SEMI-ANNUAL STATUS REPORT

Error Control Coding Techniques for Space and Satellite Communications

NASA Grant Number NAG5-557
Report Number 97-002

Principal Investigator: Daniel J. Costello, Jr.
Post-Doctoral Associate: Oscar Y. Takeshita
Graduate Assistants: Hermano A. Cabral
Jiali He
Gregory S. White

Department of Electrical Engineering
University of Notre Dame

December 1, 1997

Part I

Algebraic Interleavers for Turbo-Codes

Oscar Y. Takeshita and Daniel J. Costello, Jr.

University of Notre Dame

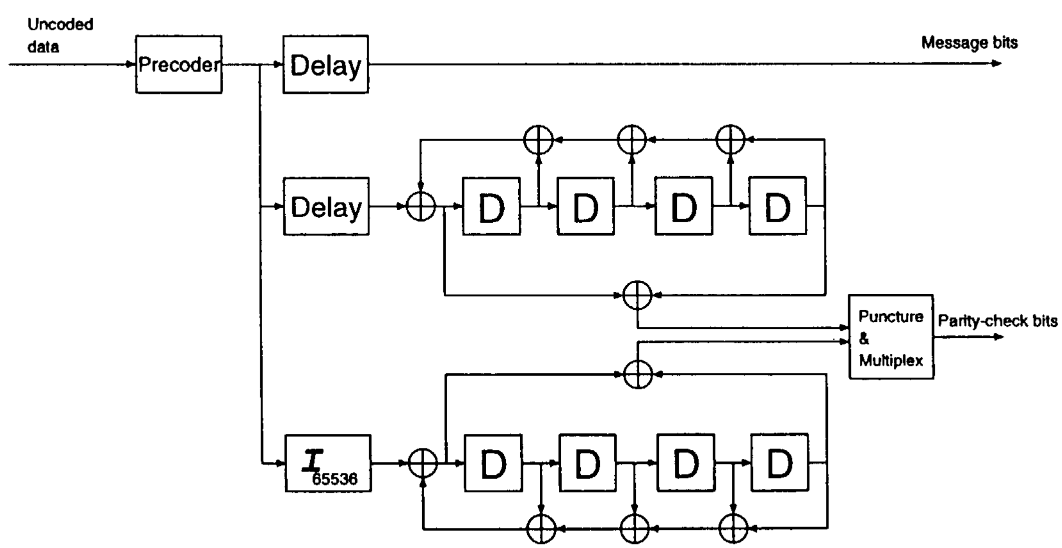
December 1, 1997

Outline

- Introduction
- Linear Interleavers
- Conclusions for Linear Interleavers
- Quadratic Interleavers
- Conclusions for Quadratic Interleavers

Introduction

- The key ingredients in the original turbo-code were a randomly chosen interleaver of a large block length of 65536 bits and a pair of systematic recursive convolutional codes (SRCC) punctured to a given rate.



- In this work we show how to construct good non-random interleavers with a performance at least as good as the average performance of several randomly chosen interleavers for a wide range of block lengths (from a few hundreds of bits to several thousands of bits). Our proposed interleavers are algebraic in nature and have very simple generation algorithms. Therefore, they have several advantages over randomly chosen interleavers, such as the possibility of analysis and a simple implementation.

- We propose two broad classes of interleavers: linear interleavers and quadratic interleavers. Simulation results indicate that for rate 1/2 turbo-codes and information block lengths less than 1000, linear interleavers are the best choice; otherwise, quadratic interleavers have better performance.

The Importance of the Cycle Length of the Component Code

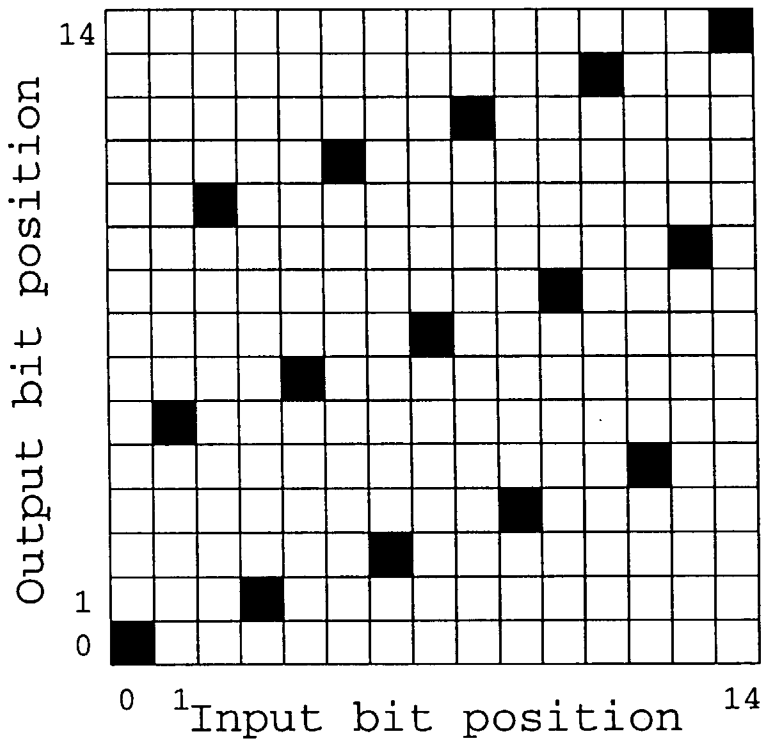
- It is known that an input sequence of the form $\overline{x}_\tau(D)D^s = (1 + D^\tau)D^s$ to an SRCC with cycle-length τ produces a low weight parity sequence. It is possible to approximately analyze the bit error rate (BER) curve of a turbo-code by inspecting how low weight parity sequences in both component codes originate.

Representations of Interleavers

- Let $\overline{x} = (x_0, x_1, \ldots, x_{N-1}) \in \{0,1\}^N$. An interleaver $\mathcal{I}_N$ maps $\overline{x}$ to a sequence $\overline{y}$ such that $\overline{y}$ is a permutation of the elements of $\overline{x}$. Let $A = \{0, \ldots, N-1\}$; $\mathcal{I}_N$ can then be defined by the one-one and onto *index mapping function* $d_{\mathcal{I}_N} : A \to A$, $d_{\mathcal{I}_N} : i \mapsto j$, $i, j \in A$, and it can be expressed as an ordered set called the *permutation vector* $\overline{\mathcal{I}}_N = [d_{\mathcal{I}_N}(0), d_{\mathcal{I}_N}(1), \ldots, d_{\mathcal{I}_N}(N-1)]$.

- Sometimes, a graphical representation of $\mathcal{I}_N$ consisting of N points in an (i,j)-plane allows a better understanding of the interleaver.

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14



- Example for a block interleaver $\mathcal{B}_{15}$ with *index mapping function* $d_{\mathcal{B}_{15}}(i) = 5i + \lfloor i/3 \rfloor \pmod{15}$.

Linear Interleavers

- Block interleavers $\mathcal{B}_N$ of size $N = m \times n$ have the *index mapping function*

  $d_{\mathcal{B}_N}(i) \equiv ni + \lfloor i/m \rfloor \pmod{N},\ 0 \le i < N.$

- To analyze a block interleaver, we use a slightly different interleaver $\mathcal{L}_N$ that "linearizes" the "floor" function $\lfloor \cdot \rfloor$ of $d_{\mathcal{B}_N}(i)$. Its *index mapping function* is

  $d_{\mathcal{L}_N}(i) \equiv ki + v \pmod{N},\ 0 \le i < N,$

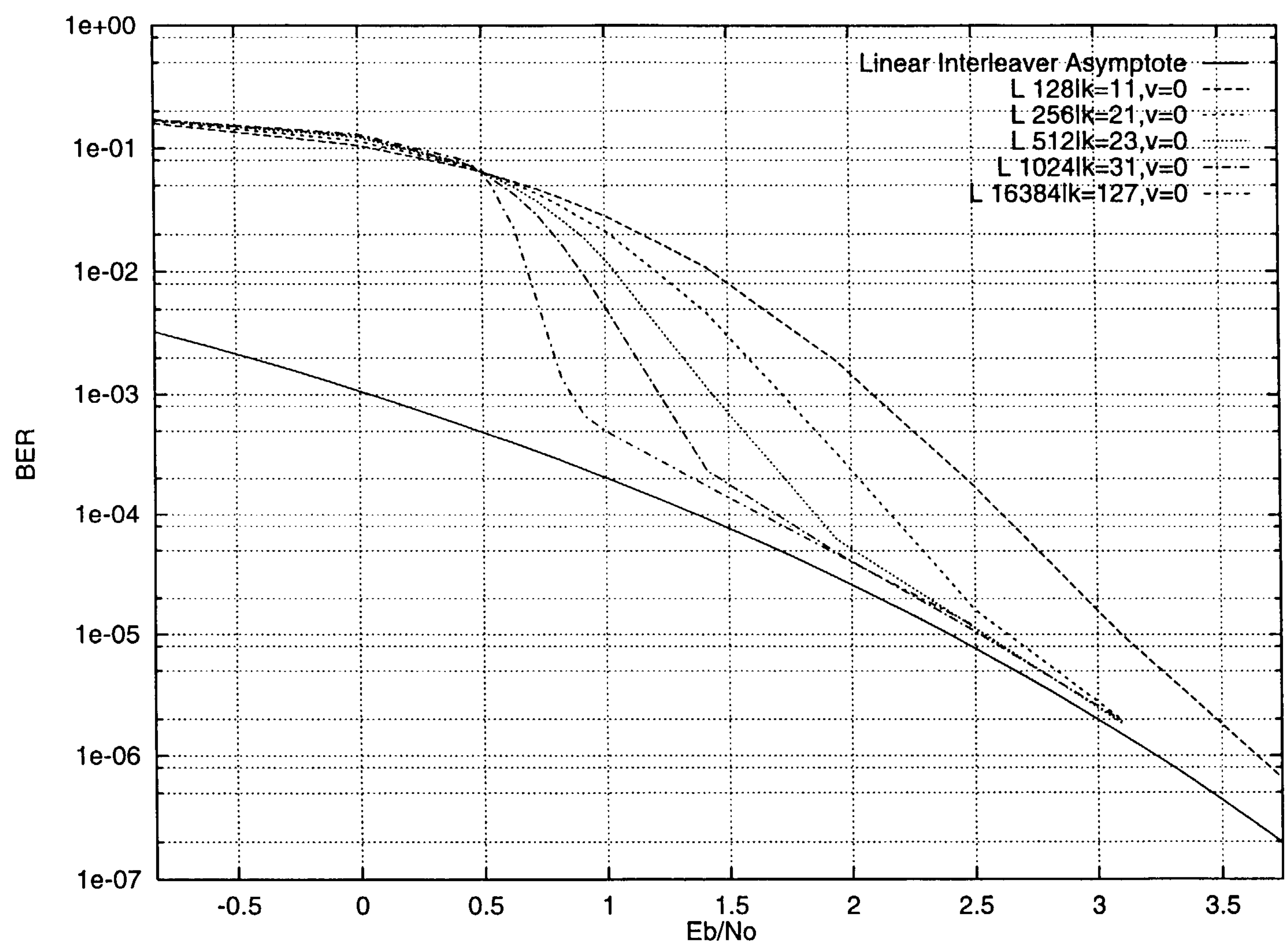
  where k (the angular coefficient) is a fixed integer relatively prime to N and v is a fixed integer. We will refer to both $\mathcal{B}_N$ and $\mathcal{L}_N$ as linear interleavers.

Bad Weight-4 Input Sequences: The Linear Asymptote

- For an interleaver $\mathcal{L}_N$, the congruence $tk \equiv \tau \pmod{N}$ is always solvable in t with a unique solution. Hence, an input sequence of the form $\overline{x}_i(D) = (1 + D^t)D^q$ to $\mathcal{L}_N$ produces the output sequence $(1 + D^\tau)D^s = \overline{x}_\tau(D)D^s$, i.e., it produces a low weight parity sequence for the second component code.

- We now give a new algebraic interpretation of the fact that a weight-4 input sequence of the form $\overline{x}_{\text{bad4}} = \overline{x}_i(D) + \overline{x}_i(D)D^\tau$ produces low weight parity sequences in both component codes (producing a codeword of weight 12 if we use the SRCC of the original turbo-code). Moreover, each of the N possible cyclic shifts of $\overline{x}_{\text{bad4}}$ produces the same result.

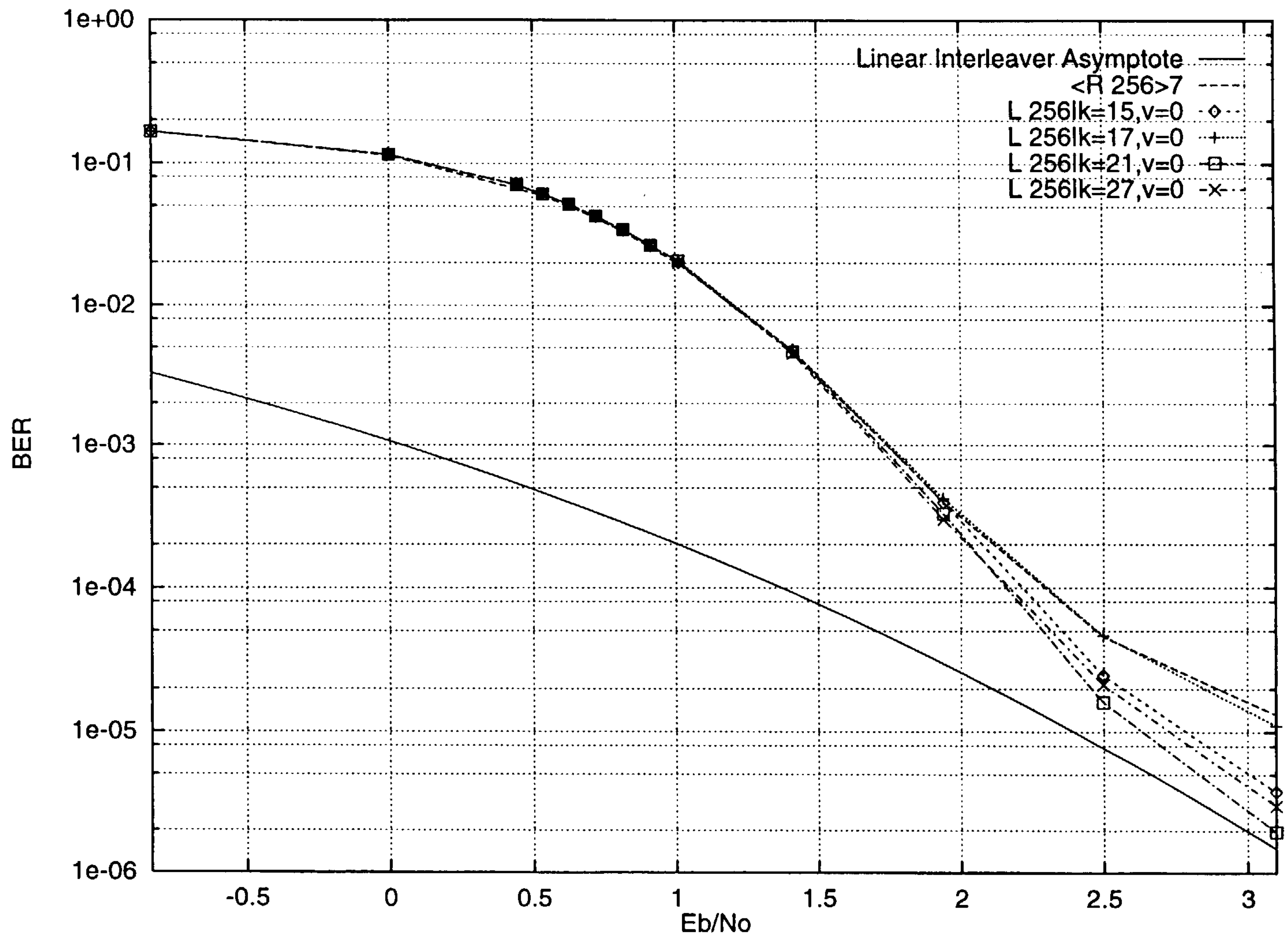
- The following asymptotic BER performance, that is invariant over the block length N and the angular coefficient k of a linear interleaver, can then be derived $P_b(e|\overline{x}_{\mathrm{bad4}}) \approx 2\mathrm{erfc}\left(\sqrt{6E_b/No}\right)$.
- In the next figure, we show the BER performance of turbo-codes with several block lengths (from 256 to 16384) using linear interleavers.



BER
1e+00
1e-01
1e-02
1e-03
1e-04
1e-05
1e-06
1e-07
-0.5
0
0.5
1
1.5
2
2.5
3
3.5
Eb/No
Linear Interleaver Asymptote
L 128lk=11,v=0
L 256lk=21,v=0
L 512lk=23,v=0
L 1024lk=31,v=0
L 16384lk=127,v=0

- The optimum value for the angular coefficient k of a linear interleaver of length N has been found to be on the order of $\sqrt{N}$, which distributes the points in the graphical representation of an interleaver throughout the (i, j)-plane.

  Note also the sensitivity of the BER curves with respect to the angular coefficients in the next figure.


Linear Interleaver Asymptote
<R 256>7
L 256lk=15,v=0
L 256lk=17,v=0
L 256lk=21,v=0
L 256lk=27,v=0
BER
1e+00
1e-01
1e-02
1e-03
1e-04
1e-05
1e-06
-0.5
0
0.5
1
1.5
2
2.5
3
Eb/No

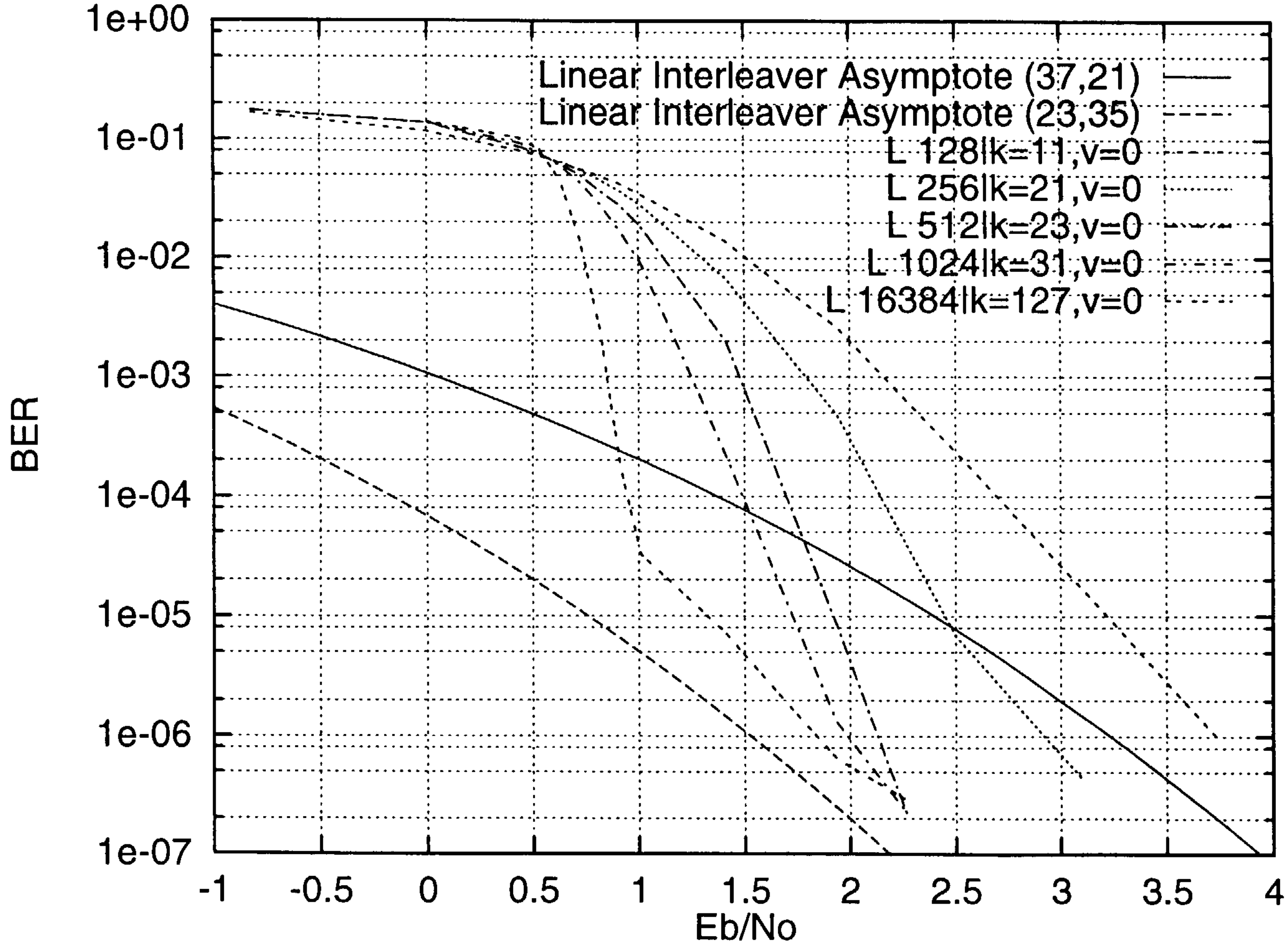
The Linear Asymptote of "Primitive" Turbo-Codes

- Several research groups have noted that using a "primitive" feedback polynomial for the SRCC helps lower the "error-floor" of a turbo-code using random interleavers. We show that a "primitive" turbo-code using linear interleavers also presents similar beneficial effects by producing a different linear asymptote:

  $$P_b(e|\overline{x}_{\text{bad4}} \text{ or } \overline{x}_{\text{bad9}}) \approx 2\text{erfc}(\sqrt{10E_b/No}) + 9/2\text{erfc}(\sqrt{9.5E_b/No}).$$

- With a "primitive" feedback polynomial, we can construct a turbo-code using a linear interleaver with block length 16384 that is only 1.1dB away from capacity at a BER of 10^{-5}, as shown in the next figure.

- The SRCC used for the "primitive" turbo-code is a 16-state (23,35) code, where 23 and 35 denote the feedback and feedforward polynomials in octal notation. The "non-primitive" turbo-code shown in the introduction, which is the original code presented by Berrou et.al., uses the 16-state (37,21) SRCC.



Linear Interleaver Asymptote (37,21)
Linear Interleaver Asymptote (23,35)
L 128lk=11,v=0
L 256lk=21,v=0
L 512lk=23,v=0
L 1024lk=31,v=0
L 16384lk=127,v=0
BER
1e+00
1e-01
1e-02
1e-03
1e-04
1e-05
1e-06
1e-07
-1
-0.5
0
0.5
1
1.5
2
2.5
3
3.5
4
Eb/No

Conclusions for Linear Interleavers

- Block and linear interleavers have been shown to be practically equivalent.
- Good linear interleavers of block length N have an angular coefficient on the order of $\sqrt{N}$. The optimum value can be easily determined by computing the weight spectrum of the turbo-code due to weight-2 input sequences.
- We have constructed a powerful non-random "primitive" turbo-code with block length 16384 that is only 1.1dB away from capacity at a BER of 10^{-5}.

Quadratic Interleavers

- We have seen that linear interleavers can be represented by the integer points of a linear curve over modular arithmetic. We have further found a remarkable result that some quadratic curves also define interleavers. We have determined that these quadratic interleavers have properties that are non-linear along the length of the interleaver. This non-linear behavior, that is shared with randomly chosen interleavers, is what makes both quadratic and randomly chosen interleavers well suited to turbo-codes with large block lengths.

Description of the Quadratic Interleavers

- We first construct a class of interleavers $\mathcal{D}_{N:CN}$ that have a block length of N. The *index mapping function* of $\mathcal{D}_{N:CN}$

$$d_{\mathcal{D}_{N:CN}} : c_m \mapsto c_{m+1} \pmod{N}, \qquad 0 \le m < N \tag{1}$$

is defined by the following algorithm.

Algorithm 0.1

$$\begin{cases} \textit{step } 1: & c_0 = 0 \\ \textit{step } 2: & c_m \equiv c_{m-1} + km \pmod{N}, \quad 0 < m < N, \\ & k \quad \textit{an odd constant.} \end{cases}$$

- Remark: The previous algorithm can also be expressed as a quadratic congruence

$$c_m \equiv \frac{km(m+1)}{2}, \; 0 \leq m < N, k \text{ an odd constant.} \qquad (2)$$

- Next we generalize of this algorithm as follows.

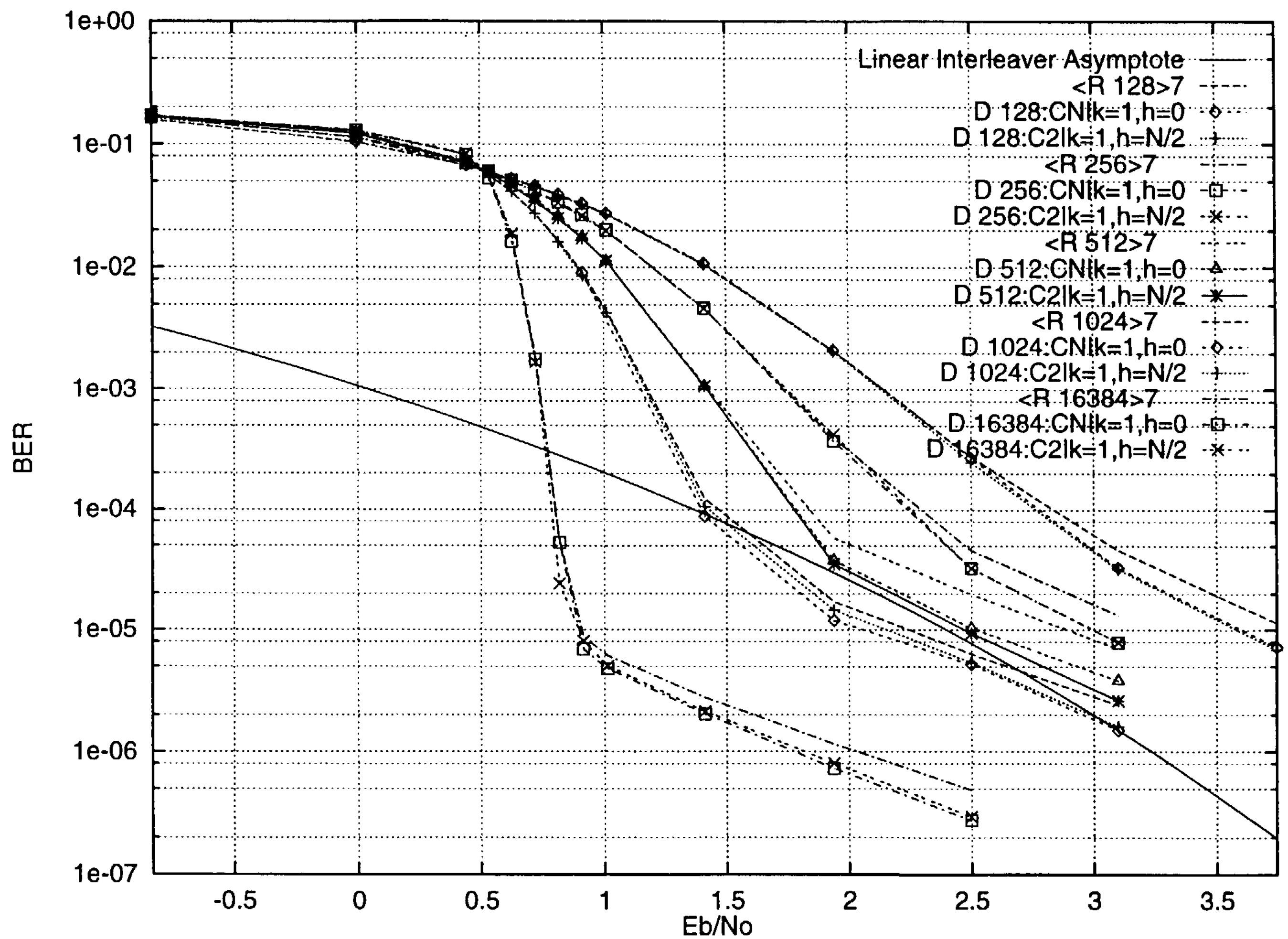
Algorithm 0.2

$$\begin{cases} \textit{step } 1: & \textit{compute the } \text{permutation vector } \overline{\mathcal{D}}_{N:CN} \\ & \textit{using } \text{Algorithm 0.1} \\ \textit{step } 2: & \textit{cyclically shift by } h \textit{ units the result of step } 1 \\ \textit{step } 3: & \textit{add a constant } v \pmod N \textit{ to each element} \\ & \textit{of the result of step } 2, \end{cases}$$

- Two special cases have been identified: $\mathcal{D}_{N:CN} = \mathcal{D}_N|_{h-v=0}$ and $\mathcal{D}_{N:C2} = \mathcal{D}_N|_{h-v=N/2}$.
- The interleavers $\mathcal{D}_{N:C2} = \mathcal{D}_N|_{h-v=N/2}$ are of particular interest because they have the property that deinterleaving is implemented using the same function. This leads to a simplification in the decoding process of a turbo-code.

Performance of the New Quadratic Interleavers Compared with Randomly Chosen Interleavers

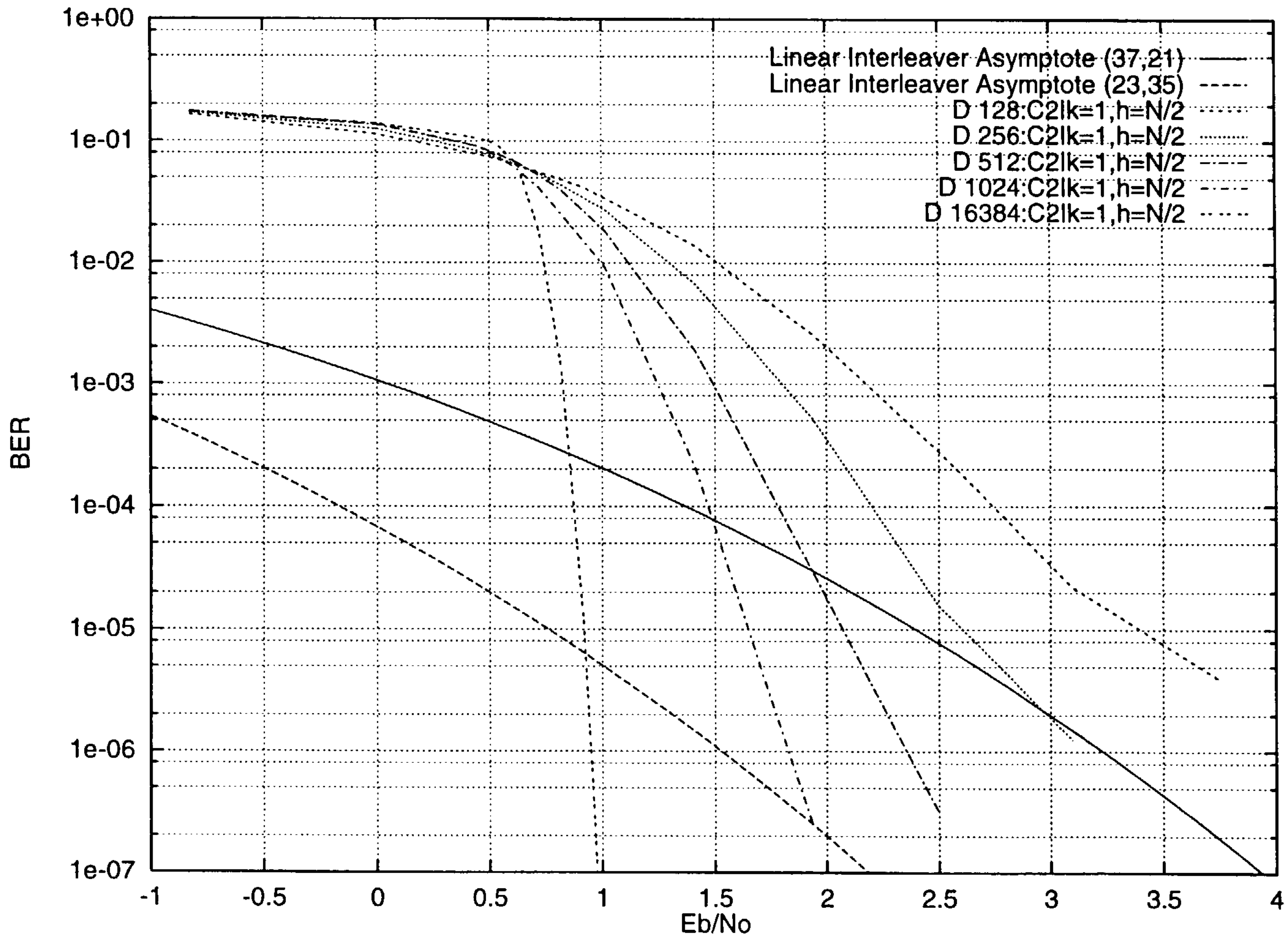
- In the next figure, we compare the BER performance of turbo-codes using quadratic interleavers against the average performance of turbo-codes using 7 randomly chosen interleavers. Note that the quadratic interleavers have performance better than the average of the randomly chosen interleavers. Moreover, the $\mathcal{D}_{N:C2}$ interleavers have excellent performance. (We used the (37,21) "non-primitive" SRCC as a component code in these simulations.)



BER
1e+00
1e-01
1e-02
1e-03
1e-04
1e-05
1e-06
1e-07
-0.5
0
0.5
1
1.5
2
2.5
3
3.5
Eb/No
Linear Interleaver Asymptote
<R 128>7
D 128:CNlk=1,h=0
D 128:C2lk=1,h=N/2
<R 256>7
D 256:CNlk=1,h=0
D 256:C2lk=1,h=N/2
<R 512>7
D 512:CNlk=1,h=0
D 512:C2lk=1,h=N/2
<R 1024>7
D 1024:CNlk=1,h=0
D 1024:C2lk=1,h=N/2
<R 16384>7
D 16384:CNlk=1,h=0
D 16384:C2lk=1,h=N/2

Performance of the New Quadratic Interleavers using "primitive" Turbo-Codes

- In the next figure, we use the new quadratic interleavers, but this time with a "primitive" turbo-code. The SRCC is a 16-state (23,35) code. The BER performance of the codes are very impressive. The flattening in the BER curve observed in the "non-primitive" turbo-code case is not observed for block lengths larger than 256.



BER
1e+00
1e-01
1e-02
1e-03
1e-04
1e-05
1e-06
1e-07
-1
-0.5
0
0.5
1
1.5
2
2.5
3
3.5
4
Eb/No
Linear Interleaver Asymptote (37,21)
Linear Interleaver Asymptote (23,35)
D 128:C2lk=1,h=N/2
D 256:C2lk=1,h=N/2
D 512:C2lk=1,h=N/2
D 1024:C2lk=1,h=N/2
D 16384:C2lk=1,h=N/2

Conclusions for Quadratic Interleavers

- A new class of quadratic interleavers have been proposed. Quadratic interleavers have non-linear properties similar to those observed in randomly chosen interleavers.
- We have shown that turbo-codes using quadratic interleavers have a BER performance superior to the average BER performance of turbo-codes using randomly chosen interleavers.
- Quadratic interleavers have a very simple generation algorithm. This means a significant advantage over other known interleaver generation algorithms.

Part II

Iterative Sequential Decoding for Trellis Coded Modulation

Hermano A. Cabral and Daniel J. Costello, Jr.

University of Notre Dame

December 1, 1997

Introduction

- Iterative decoding has become a powerful tool which enables information transmission at rates close to capacity. The best example of such an achievement is turbo-coding, which achieves surprisingly good performance on a binary-input AWGN channel at low Signal-to-Noise Ratios (SNR's).
- Another iterative scheme which can achieve good performance is *Bootstrap Hybrid Decoding* (BHD), which uses extra parity checks to improve the likelihood function used by a sequential decoder, thereby obtaining better computational behavior compared to normal sequential decoding.

- This report discusses an extension of the BHD scheme to Trellis-Coded Modulation (TCM) and compares its performance to a similar extension using turbo-codes.

Sequential Decoding

- It is well known that there exist large constraint length convolutional codes yielding very low Bit Error Rates (BER's) at rates arbitrarily close to capacity when paired with an optimum decoder. The problem is that such decoders are not practical due to their large complexity.

- For such codes, a suboptimum decoding algorithm, like sequential decoding, is attractive, since its complexity is almost independent of the constraint length. Its slightly suboptimum performance can be overcome by using a larger constraint length code, and therefore a BER close to zero can be achieved.

- The problem with sequential decoding is its probabilistic behavior, since its computational load depends on the noise and is therefore a random variable. In particular, for rates above the computational cut-off rate R_0, its computational load has an infinite expected value, and hence R_0 is considered the practical limit for sequential decoding.

The Bootstrap Hybrid Decoding Idea

- BHD, however, has a larger effective cut-off rate, and thus allows sequential decoding to operate at rates closer to capacity. It works by taking a set of $m-1$ codewords from a convolutional code and adding redundancy in the form of a new codeword such that the sum of all m codewords over the binary field $\mathrm{GF}(2)$ is the all-zero sequence. This redundancy can then be used at the decoder to improve the sequential decoder's likelihood function.

- The decoding scheme becomes iterative by using the estimated sequences provided by the sequential decoder to update the likelihood function adjustment, since, with high probability, these estimates are correct.

Generalizing BHD

- The BHD idea can be applied to both BPSK and larger signal constellations. However, the metric adjustment has been derived only for the BPSK case, where it depends on one sequence, called the *channel state stream*, calculated from all the received sequences.
- For larger constellations, the adjustment depends on how the redundancy is introduced. Assuming a TCM scheme using an 8-PSK constellation and a rate 2/3 convolutional code, the extra redundant sequence can be derived by treating the codewords from the convolutional code as vectors of binary 3-tuples and adding them componentwise over $\mathrm{GF}(8)$. The resulting sequence is still a codeword, and therefore can be mapped exactly like the other codewords.

- For the sake of exposition, we can view the m codewords as a matrix, where each row is a codeword and each codeword is a sequence of symbols from $\mathrm{GF}(8)$. If a codeword has L branches (including the tail), then the above matrix has dimension $m \times L$.
- Using this representation, an $m \times L$ matrix A with elements from $\mathrm{GF}(8)$ represents a valid set of m codewords if, and only if, the sum of the m rows is identically zero. Moreover, if we partition the set of rows into two subsets and compute the sum of the rows in each of these subsets, the above condition implies that the two sums are equal.

- We now assume, without loss of generality, that the last l rows have been decoded, and hence that $m' = m - l$ rows remain to be decoded. Then, by partitioning the set of rows into two subsets, one containing the first m' rows and the other containing the l decoded rows, the above condition implies that the sum of the rows in the first subset is equal to the sum of rows in the second subset.

- Call the decoded subset sum, which is L symbols long, the channel state stream $S = (S_1, S_2, \ldots, S_L)$, and assume that row m' is next to be decoded.

- If $\mathbf{x}_i^{(m')} = (x_{i,1}, x_{i,2}, \ldots, x_{i,m'})$ denotes the vector consisting of the first m' elements of the i-th column of A, and $\mathbf{y}_i^{(m')} = (y_{i,1}, y_{i,2}, \ldots, y_{i,m'})$ is the corresponding received vector, then the likelihood function for the m'-th row should be

$$\lambda_i(m') = \log \frac{p(\mathbf{y}_i^{(m')} \mid x_{i,m'}, S_i)}{p(\mathbf{y}_i^{(m')} \mid S_i)} - R, \quad i = 1 \ldots, L \qquad (1)$$

where R is the code rate in bits per channel use.

- Let $V_s^{(m')}$, $s \in \mathrm{GF}(8)$, be the set of all vectors $\mathbf{x}^{(m')} = (x_1, \ldots, x_{m'})$ over $\mathrm{GF}(8)$ such that $\sum x_j = s$, where the sum is carried out over $\mathrm{GF}(8)$. Then the sets $V_s^{(m')}$, $s \in \mathrm{GF}(8)$, form a partition of the set of all m'-tuples over $\mathrm{GF}(8)$, and each set has cardinality $8^{(m'-1)}$.

- We then have

$$\begin{aligned} p(\mathbf{y}_i^{(m')} \mid S_i) &= \sum_{\mathbf{x}^{(m')} \in V_{S_i}^{(m')}} p(\mathbf{y}_i^{(m')} \mid \mathbf{x}^{(m')}) p(\mathbf{x}^{(m')}) \\ &= \frac{1}{8^{(m'-1)}} \sum_{\mathbf{x}^{(m')} \in V_{S_i}^{(m')}} \prod_{j=1}^{m'} p(y_{i,j} \mid x_j), \end{aligned}$$

since the channel is memoryless.

- The probability $p(\mathbf{y}_i^{(m')} \mid x_{i,m'}, S_i)$ can be found in a similar way by noting that if $x_{i,m'}$ is given, then the set of unknown $x_{i,j}$ in the i-th column is $(x_{i,1}, \ldots, x_{i,m'-1})$ and the channel state is $S_i \oplus x_{i,m'}$. This gives

$$
\begin{aligned}
& p(\mathbf{y}_i^{(m')} \mid x_{i,m'}, S_i) = \\
&= \sum_{\mathbf{x}^{(m'-1)} \in V_{S_i \oplus x_{i,m'}}^{(m'-1)}} p(\mathbf{y}_i^{(m')} \mid \mathbf{x}^{(m'-1)}, x_{i,m'}) p(\mathbf{x}^{(m'-1)}) \\
&= \frac{p(y_{i,m'} \mid x_{i,m'})}{8^{(m'-2)}} \sum_{\mathbf{x}^{(m'-1)} \in V_{S_i \oplus x_{i,m'}}^{(m'-1)}} \prod_{j=1}^{m'-1} p(y_{i,j} \mid x_j)
\end{aligned}
$$

- Defining $\Delta_{i,s}^{(m')}$, $s \in \mathrm{GF}(8)$, as

$$\Delta_{i,s}^{(m')} = \sum_{\mathbf{x}^{(m')} \in V_s^{(m')}} \prod_{j=1}^{m'} p(y_{i,j} \mid x_j)$$

allows us to rewrite the above probabilities as,

$$p(\mathbf{y}_i^{(m')} \mid S_i) = \frac{\Delta_{i,S_i}^{(m')}}{8^{(m'-1)}} \qquad (2)$$

and

$$p(\mathbf{y}_i^{(m')} \mid x_{i,m'}, S_i) = \frac{p(y_{i,m'} \mid x_{i,m'})}{8^{(m'-2)}} \Delta_{i,S_i \oplus x_{i,m'}}^{(m'-1)} \qquad (3)$$

- Let now C be a subset of $\mathrm{GF}(8)$ and $\bar{C}$ be its complement, and consider the following product,

$$\prod_{j=1}^{m'} \left(\sum_{x \in C} p(y_{i,j} \mid x) - \sum_{x \in \bar{C}} p(y_{i,j} \mid x) \right) . \tag{4}$$

- This product can be expanded into the following sum,

$$\sum_{x_1 \in \mathrm{GF}(8)} \cdots \sum_{x_{m'} \in \mathrm{GF}(8)} (-1)^{f(x_1, \ldots, x_{m'})} \prod_{j=1}^{m'} p(y_{i,j} \mid x_j), \tag{5}$$

where $f(x_1, \ldots, x_{m'}) \triangleq f(\mathbf{x}^{(m')})$ is either 0 or 1.

- Notice that every vector $\mathbf{x}^{(m')} = (x_1, \ldots, x_{m'})$ in the set of all m'-tuples over $\mathrm{GF}(8)$ is represented once, and only once, in the above sum, and that the corresponding sign will be negative (that is, $f(\mathbf{x}^{(m')}) = 1$) if, and only if, an odd number of the x_j's are in $\bar{C}$ ($f(\mathbf{x}^{(m')}) = 0$ otherwise).

- If $C = \{(000), c_1, c_2, c_1 \oplus c_2\}$, where c_1 and c_2 are distinct non-zero elements of $\mathrm{GF}(8)$, is a rate 2/3 binary linear block code, then $\bar{C}$ is its non-zero coset and has the property that the sum of any even number of elements $x_j \in \bar{C}$ is in C, and the sum of any odd number of elements $x_j \in \bar{C}$ is in $\bar{C}$.

- In this case, $f(\mathbf{x}^{(m')}) = 0$ if, and only if, $\sum_{j=1}^{m'} x_j \in C$. If U denotes the set of all m'-tuples s over $\mathrm{GF}(8)$ such that $\sum_{j=1}^{(m')} x_j \in C$ then (5) can be rewritten as

$$\sum_{\mathbf{x}^{(m')} \in U} \prod_{j=1}^{m'} p(y_{i,j} \mid x_j) - \sum_{\mathbf{x}^{(m')} \notin U} \prod_{j=1}^{m'} p(y_{i,j} \mid x_j)$$

- But from the definition of $V_s^{(m')}$, we have $U = V_{(000)}^{(m')} \cup V_{c_1}^{(m')} \cup V_{c_2}^{(m')} \cup V_{c_1 \oplus c_2}^{(m')}$, and therefore the above sum becomes

$$(\Delta_{i,(000)}^{m'} + \Delta_{i,c_1}^{m'} + \Delta_{i,c_2}^{m'} + \Delta_{i,c_1 \oplus c_2}^{m'}) - \\ (\Delta_{i,e_1}^{m'} + \Delta_{i,e_1 \oplus c_1}^{m'} + \Delta_{i,e_1 \oplus c_2}^{m'} + \Delta_{i,e_1 \oplus c_1 \oplus c_2}^{m'}), \qquad (6)$$

where e_1 is the coset leader.

- There are seven different rate 2/3 codes that can be chosen, namely,

Code	c_1	c_2
C_1	001	010
C_2	001	100
C_3	001	110
C_4	010	100

Code	c_1	c_2
C_5	010	101
C_6	100	011
C_7	011	110

each yielding a different value for (6). Call each of these values $E_{i,l}^{(m')}$, $l = 1, \ldots, 7$. An additional value can be obtained by making $C = \mathrm{GF}(8)$ in (4) (i.e., $f(\mathbf{x}^{(m')}) = 0$ for all $\mathbf{x}^{(m')}$). Call this sum $E_{i,0}^{(m')}$.

We can then write the following linear system of equations,

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & 1 & -1 & -1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{pmatrix} \begin{pmatrix} \Delta_{i,0}^{(m')} \\ \Delta_{i,1}^{(m')} \\ \Delta_{i,2}^{(m')} \\ \Delta_{i,3}^{(m')} \\ \Delta_{i,4}^{(m')} \\ \Delta_{i,5}^{(m')} \\ \Delta_{i,6}^{(m')} \\ \Delta_{i,7}^{(m')} \end{pmatrix} = \begin{pmatrix} E_{i,0}^{(m')} \\ E_{i,1}^{(m')} \\ E_{i,2}^{(m')} \\ E_{i,3}^{(m')} \\ E_{i,4}^{(m')} \\ E_{i,5}^{(m')} \\ E_{i,6}^{(m')} \\ E_{i,7}^{(m')} \end{pmatrix},$$

where

$$(000) \triangleq 0, (001) \triangleq 1, \ldots, (111) \triangleq 7.$$

- The above system can be written as

$$\Omega \boldsymbol{\Delta}_i = \boldsymbol{E}_i.$$

- The vector $\boldsymbol{E}_i$ can be found using (4) and so, since Ω is invertible, $\boldsymbol{\Delta}_i$ is known and can be used to compute (2) and (3), and therefore the likelihood function (1) is

$$\lambda_i(m') = \log(y_{i,m'} \mid x_{i,m'}) + \log \frac{\Delta_{i,S_i \oplus x_{i,m'}}^{(m'-1)}}{\Delta_{i,S_i}^{(m')}} + 3 - R.$$

- The generalization to the j-th stream is straightforward, with the likelihood function being

$$\lambda_i(j) = \log(y_{i,j} \mid x_{i,j}) + \log \frac{\Delta_{i,S_i \oplus x_{i,j}}^{(m'-1)}}{\Delta_{i,S_i}^{(m')}} + 3 - R.$$

- If the j-th stream is successfully decoded, then the channel state stream is updated,

$$(S_1 \oplus x_{1,j}, S_2 \oplus x_{2,j}, \ldots, S_L \oplus x_{L,j}),$$

  and the vectors $\boldsymbol{E}_i,\ i = 1, \ldots, L$, are recomputed. This makes the whole process iterative.

- The decoding process is finished when either all streams have been successfully decoded, or when some computational limit is exceeded, in which case the frame is declared an erasure and whatever streams were decoded are released to the user.

Conclusions

- An extension of the Bootstrap Hybrid Decoding scheme to TCM was presented, detailing how to obtain the metric adjustments for the sequential decoders, as well as their updates based on successful decoder estimates.
- A simulation program is being developed to assess the performance of the BHD/TCM scheme and to compare it with the performance of a similar scheme using turbo-codes.
- Analysis of the computational behavior is also being performed to determine the theoretical limits of this scheme.

Part III

Some new Turbo coding schemes

Jiali He and Daniel J. Costello, Jr.

University of Notre Dame

December 1, 1997

- In our previous report, we introduced several concatenation schemes using Turbo codes as the inner code in an attempt to achieve all or some of the following objectives: (1) eliminate (or reduce) the error floor; (2) achieve similar performance with less decoding effort; and (3) achieve similar performance with less decoding delay.

- These schemes included: using an outer RS code or BCH code to correct the small number of errors after Turbo decoding; using an interleaver to spread the symbol errors (for an outer RS code) or bit errors (for an outer BCH code) in one block to several blocks, thus making the errors more random and generating more correctable blocks; using the soft-output information from the inner decoder to declare some symbols or bits as erasures for an outer errors and erasures decoder; using feedback from the outer decoder to help the inner Turbo decoder; and using an outer CRC for 'data integrity verification' and to reduce average decoding complexity.

- **However, these schemes with outer block codes have several disadvantages:**

  --The outer decoder begins decoding only after the inner Turbo code has finished several decoding iterations, i.e., most of the time, the outer decoder is idle.

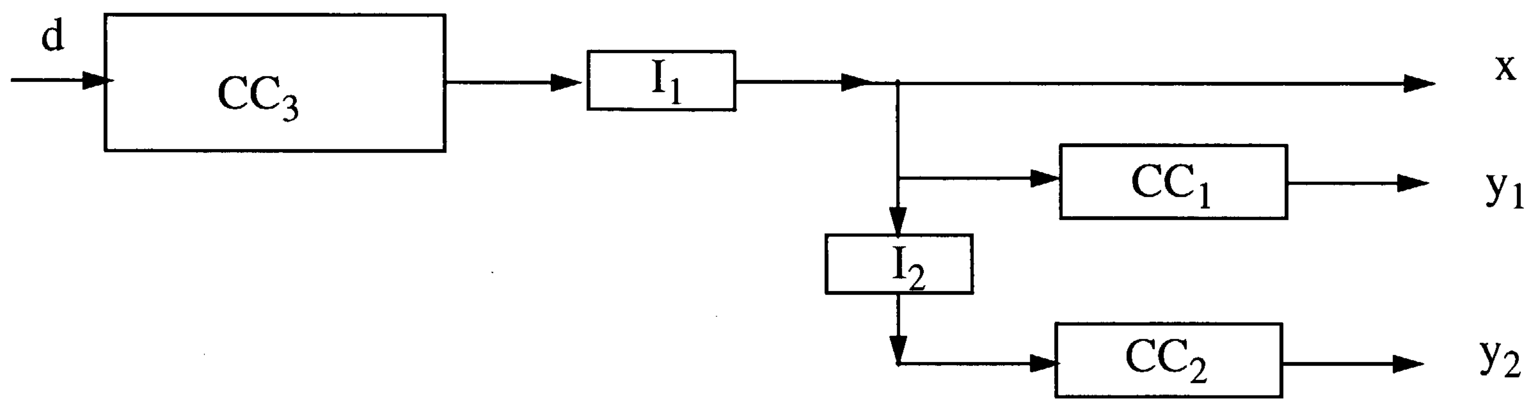
  --If the outer decoder uses a hard-decision decoding algorithm, or errors and erasures decoding, the soft information from the inner decoder output is not fully utilized.

  --The Turbo decoder output is bursty, so the number of errors is often beyond the error correcting capability of the outer code.

- **It would be desirable to have a concatenation scheme in which the outer decoder fully exploits the soft information from the inner decoder and provides positive feedback to the inner decoder throughout the entire iterative decoding process.**

- Convolutional outer codes can make full use of the soft information from the inner decoder and interact fully with the inner Turbo codes through the iterative decoding process, and thus can be used as the outer code. This results in a combination of serial and parallel concatenation.
- Parallel concatenated codes are designed for near-capacity performance at moderate BER's, while serial concatenated codes are designed for almost error-free performance. A proper combination of serial and parallel concatenation should maintain the advantages of both.
- It is important in such a hybrid coding scheme that the Turbo decoding principle (i.e., iterative decoding) be applied so that the outer decoder works together with the inner decoder.

- **The new hybrid encoding structure:**



- **Exact weight distributions at short block lengths are calculated to examine the potential of this new coding schemes.**

- **Using a rate 3/4 outer code and a rate 2/3 inner Turbo code as an example, we obtained the following weight distribution for very short block length (N= 27). (The outer code is 8-state and non-recursive, and the Turbo code is the original 'non-primitive' (37,21) code with additional puncturing.)**

Weight	Multiplicity
0	1
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	1
9	3
10	3
11	11
12	20
13	86
14	211
15	614
16	1727
17	4442
18	11045
19	25575

- **Weight distribution of the hybrid coding scheme. (Parameters are same as the previous page except that the inner code is the 'primitive' (23, 35) Turbo code.)**

Weight	Multiplicity
0	1
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	3
11	6
12	14
13	60
14	150
15	452
16	1356
17	3507
18	8839
19	21247
20	48161

- **Weight distribution of the standard rate 1/2 Turbo code** (N=27). **(The component code is the original 'non-primitive' (37,21) code.)**

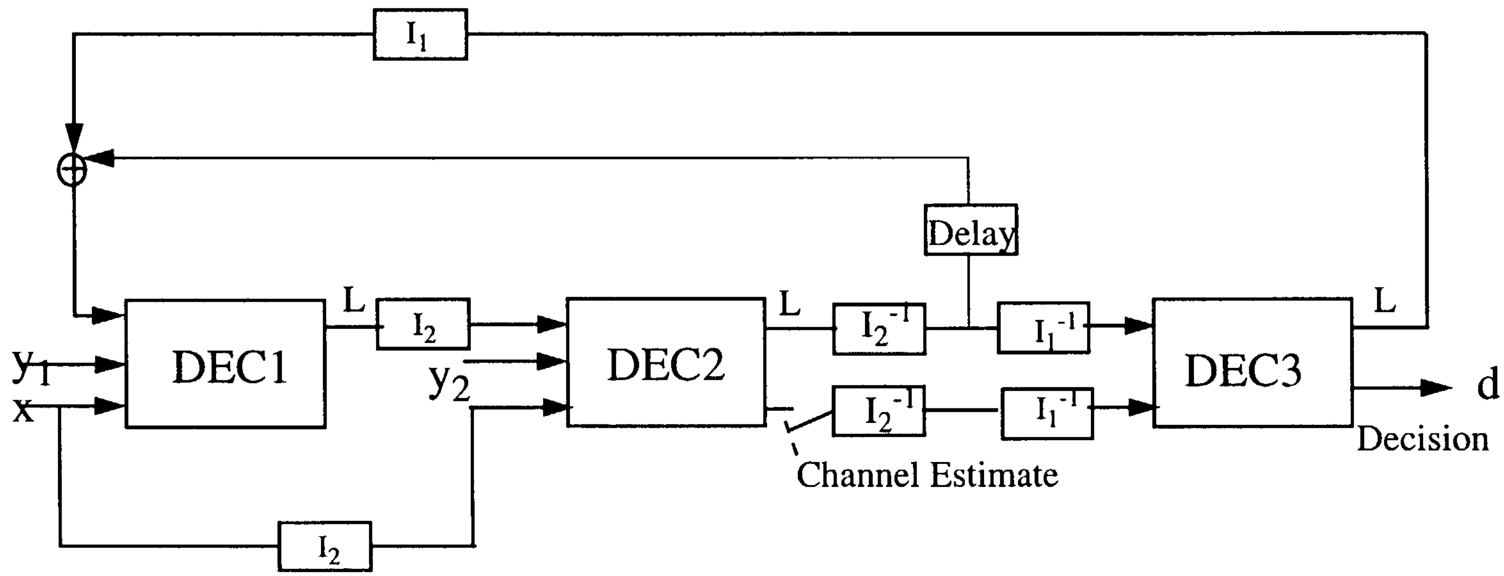
Weight	Multiplicity
0	1
1	0
2	0
3	0
4	1
5	1
6	6
7	5
8	18
9	35
10	100
11	203
12	587
13	1352
14	3404
15	7938
16	19219
17	42943
18	91451
19	187289
20	362193
21	666783

- **Weight distribution of the standard rate 1/2 Turbo code.** (N=27, **but now the component code is the 'primitive' (23,35) code.)**

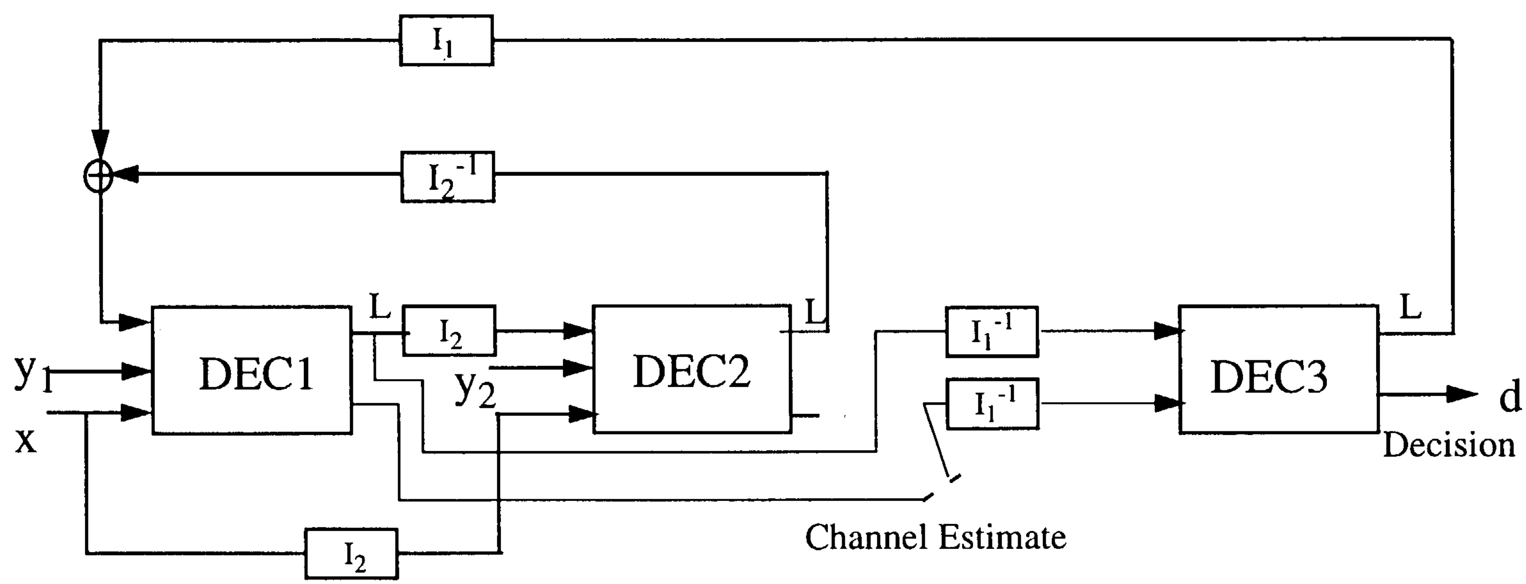
Weight	Multiplicity
0	1
1	0
2	0
3	0
4	0
5	0
6	1
7	0
8	14
9	19
10	53
11	142
12	364
13	943
14	2488
15	6458
16	15889
17	37458
18	82773
19	174154
20	345741
21	649011

- **As can be seen from comparing the two weight distributions, the hybrid structure has a ‘thinner’ distance spectrum and a larger minimum distance, as expected, since the outer code ‘filters out’ the bad input sequences for the inner Turbo code.**
- **It is expected that, at longer block lengths, this hybrid coding structure will have the characteristics of both the parallel structure (a ‘thin’ overall distance spectrum) and the serial structure (a much larger minimum distance).**
- **In all examples, the weight distributions shown are ‘averaged’ over several random interleavers.**
- **Several decoding structures based on the Turbo decoding principle have been proposed.**

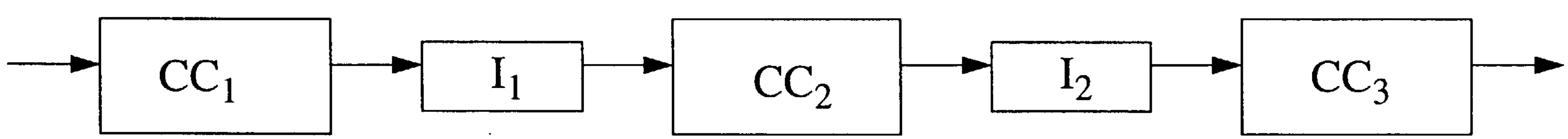
- **A possible decoding structure for the hybrid coding scheme:**



- **Another possible decoding structure for the hybrid coding scheme(DEC2 and DEC3 can work simultaneously):**



- **In our previous report, we investigated the serial concatenation of two convolutional codes.**
- **We have now extended this to a serial concatenation of three convolutional codes. Due to the serial structure, this coding scheme should give very good performance and may be useful in very low BER applications.**
- **The encoding structure (only the outer code is terminated):**



- **Weight distribution for a serial concatenation of three codes, with rates 1/2, 2/3, and 3/4 respectively (the overall rate is 1/4 and** N=27). **(The codes have 4, 4, and 8 states, and are all non-recursive.)**

Weight	Multiplicity
0	1
1	0
2	0
.	.
.	.
26	0
27	0
28	2
29	2
30	10
31	18
32	58
33	177
34	411
35	933
36	2069
37	4664
38	9402
39	18661
40	35545
41	66176

- **Weight distribution of a parallel concatenation of three convolutional codes (with overall rate 1/4 and N= 27). (The component codes are the original 'non-primitive' (37,21) recursive codes.)**

Weight	Multiplicity
0	1
1	0
2	0
.	.
.	.
14	0
15	0
16	3
17	1
18	0
19	1
20	1
21	4
22	2
23	5
24	10
25	10
26	16
27	27
28	30
29	64

- **Weight distribution of a parallel concatenation of three convolutional codes (with overall rate 1/4 and N=27). (The component codes are the 'primitive' (23,35) recursive codes.)**

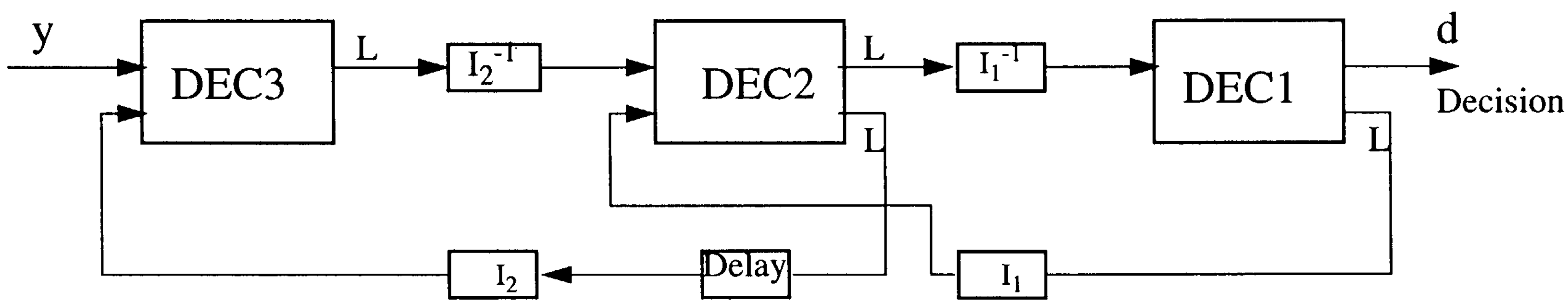
Weight	Multiplicity
0	1
1	0
2	0
.	.
.	.
20	0
21	0
22	1
23	4
24	8
25	4
26	7
27	14
28	20
29	24
30	55
31	95
32	155
33	279
34	411
35	789

- **Weight distribution of a serial concatenation of two convolutional codes, with rates 1/2 and 1/2, respectively (the overall rate is 1/4 and** N= 27). **(Both codes are 8-state and non-recursive.)**

Weight	Multiplicity
0	1
1	0
2	0
.	.
.	.
16	0
17	0
18	1
19	0
20	1
21	1
22	0
23	3
24	5
25	6
26	8
27	9
28	11
29	26
30	52
31	81

- Compared to the schemes with three parallel concatenated codes and two serial concatenated codes, the weight distribution of the three serial concatenated codes is very encouraging, especially considering the fact that the three component codes are quite simple, with 4, 4, and 8 states, respectively. The best rate 1/4 convolutional codes with free distance around 28, for example, have $2^9 \sim 2^{10}$ states.
- It has been observed that, at very short block lengths, the weight distribution of the triple serial concatenation scheme with two recursive inner codes is a little worse than with the equivalent non-recursive inner codes. However, at larger block lengths, recursive codes should be used to maximize the 'interleaver gain'.

- **A possible decoding structure for the serial concatenation of three codes:**



Summary

- In an attempt to lower the error floor of Turbo codes, we have proposed a hybrid concatenated coding scheme with an outer convolutional code and an inner Turbo code. Compared to standard Turbo codes, the weight distribution of this hybrid scheme is superior.
- Also, the weight distribution of a triple serial concatenation is very encouraging, at least compared to a parallel concatenation of three codes and a serial concatenation of two codes with the same overall rate and similar decoding complexity.
- Although the weight distributions calculated are only for very short block lengths, we expect these two new coding schemes will maintain their advantage for larger block lengths. Analysis will be required to determine the behavior of these codes for large block lengths.
- Several decoding structures for these coding schemes have been proposed and are being investigated. Iterative decoding is employed in such a way that the component codes interact fully with each other.

Part IV

Turbo Coding With Differentially Coherent and Non-Coherent Modulation in Non-Fading and Fading Channels

Gregory S. White and Daniel J. Costello, Jr.

University of Notre Dame

December 1, 1997

- Most research has focused on binary coherent modulation
  - Little activity in applying Turbo coding techniques to M-ary non-coherent modulation

- Turbo coding with differentially coherent or non-coherent modulation could provide an attractive modulation/coding solution for many systems and applications
  - Phase acquisition and tracking not required
  - Robust fading performance with channel interleaving

- Emphasis is on modulation / Turbo coding schemes characterized by
  - Robustness in Rayleigh fading
  - SOVA decoding of constituent codes
  - Power and bandwidth efficiency
  - Small interleaver length and low number of iterations
    - Packet communications
  - M-ary DPSK and FSK with binary Turbo coding
    - Constant envelope for hard limited channels

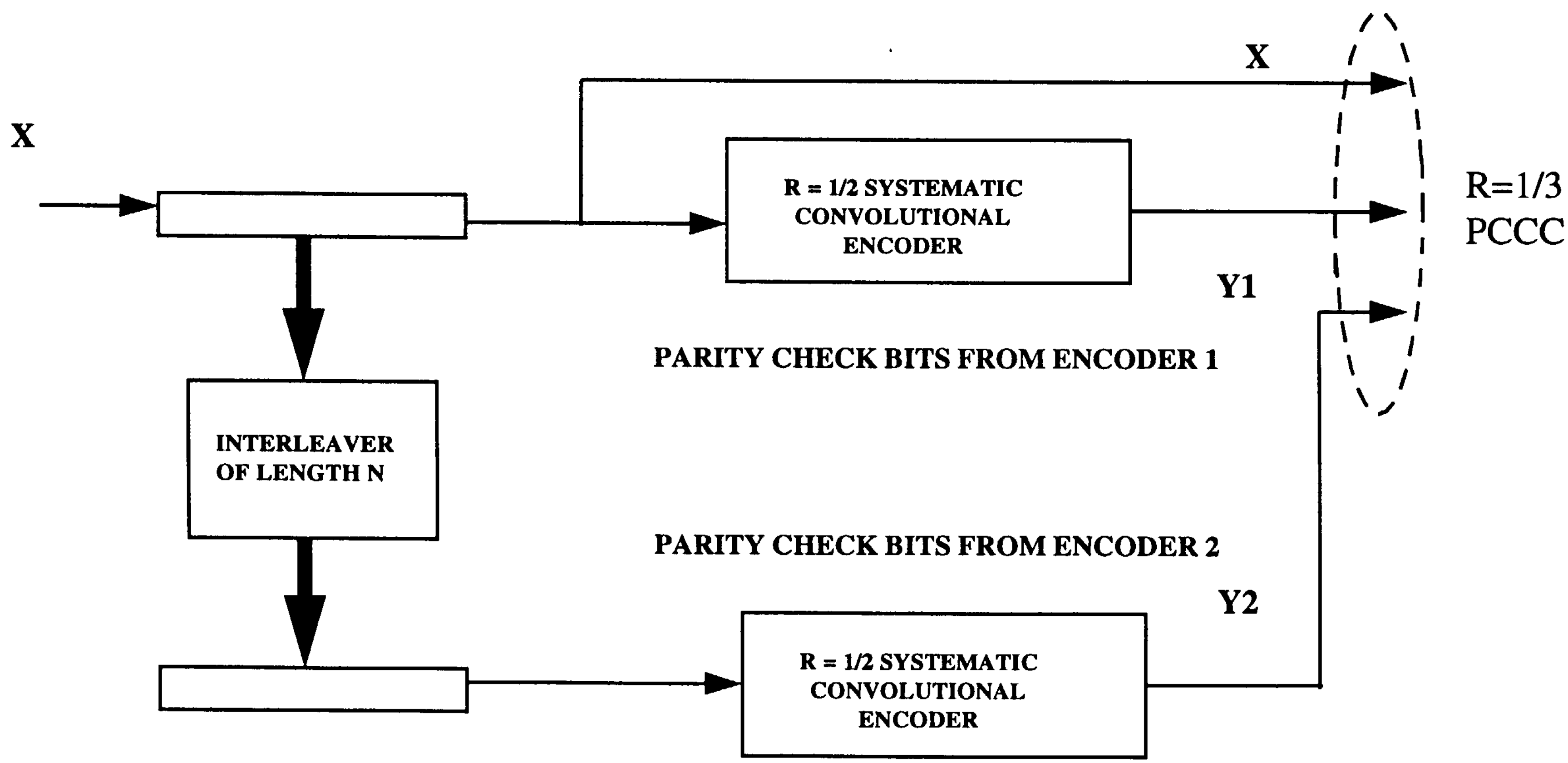
Modulation / Coding / Channel Parameters

- M-ary DPSK and FSK for M = 2, 4, and 8
  - Throughput of 0.33 to 1.50 bits / symbol
  - Channel interleaving for fading channel
- R = 1/2 systematic recursive convolutional constituent codes
  - R = 1/3 PCCC
  - R = 1/2 PCCC (constituent codes are punctured to R = 2/3)
  - 4, 8, and 16 states, $G = (1, h_1 / h_0)$

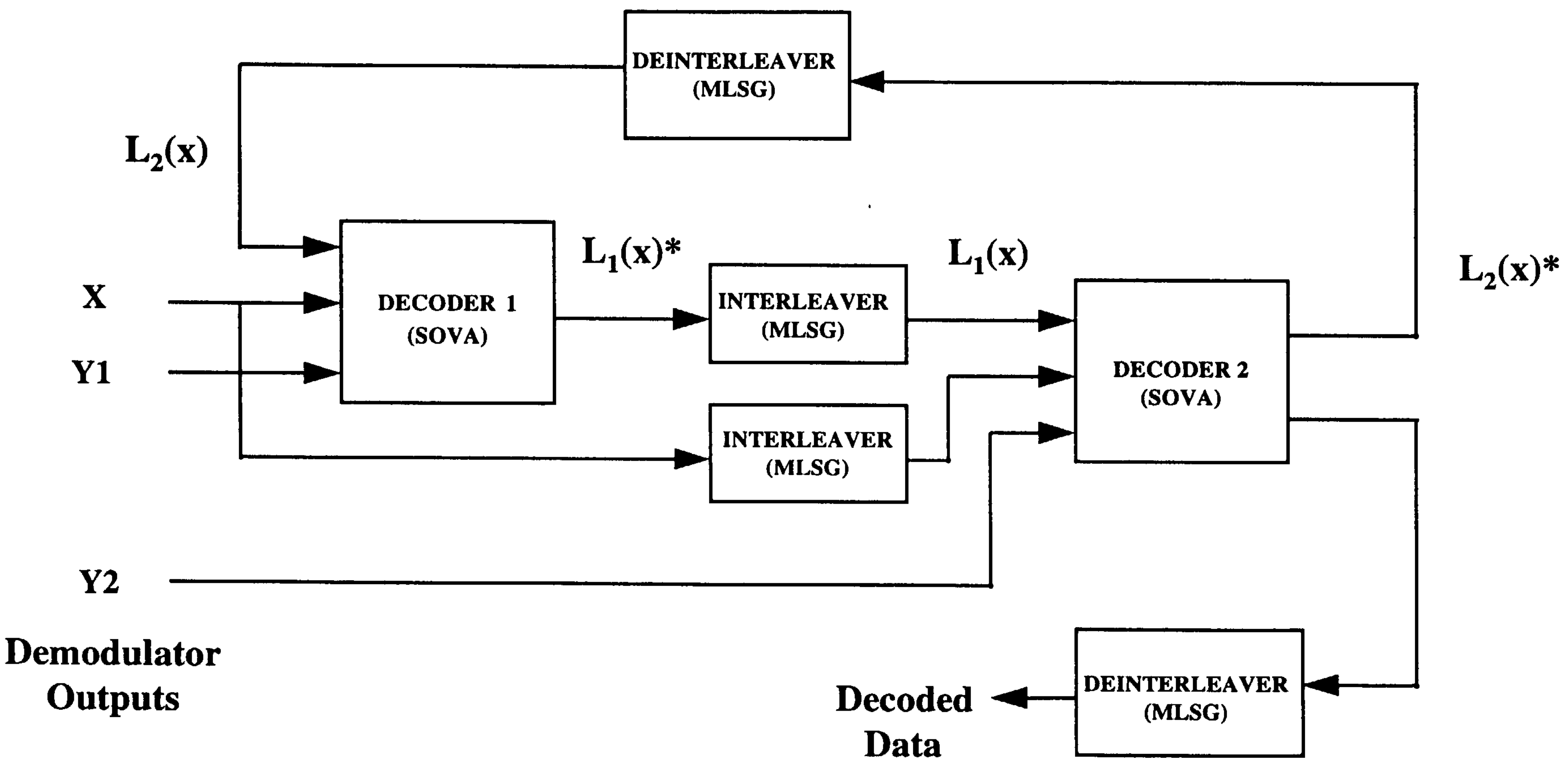
	R=1/2 CC					R=2/3 CC (punctured)				
m	h_0	h_1	d_2	d_3	d_{free}	h_0	h_1	d_2	d_3	d_{free}
2	7	5	6	5	5	7	5	4	3	3
3	15	17	8	7	6	13	15	5	4	4
4	31	37	12	8	6	23	37	7	4	4

- Additive White Gaussian Noise (AWGN); non-fading
- Additive White Gaussian Noise (AWGN); fading
  - Jakes Rayleigh fading model; slow (t_0 = 50 symbols) and fast fading (t_0 = 5 symbols)
- Small data packet lengths
  - Interleaver sizes of 511, 1023, 2047 bits

Parallel Concatenated Convolutional Code Structure (R = 1/3 example)

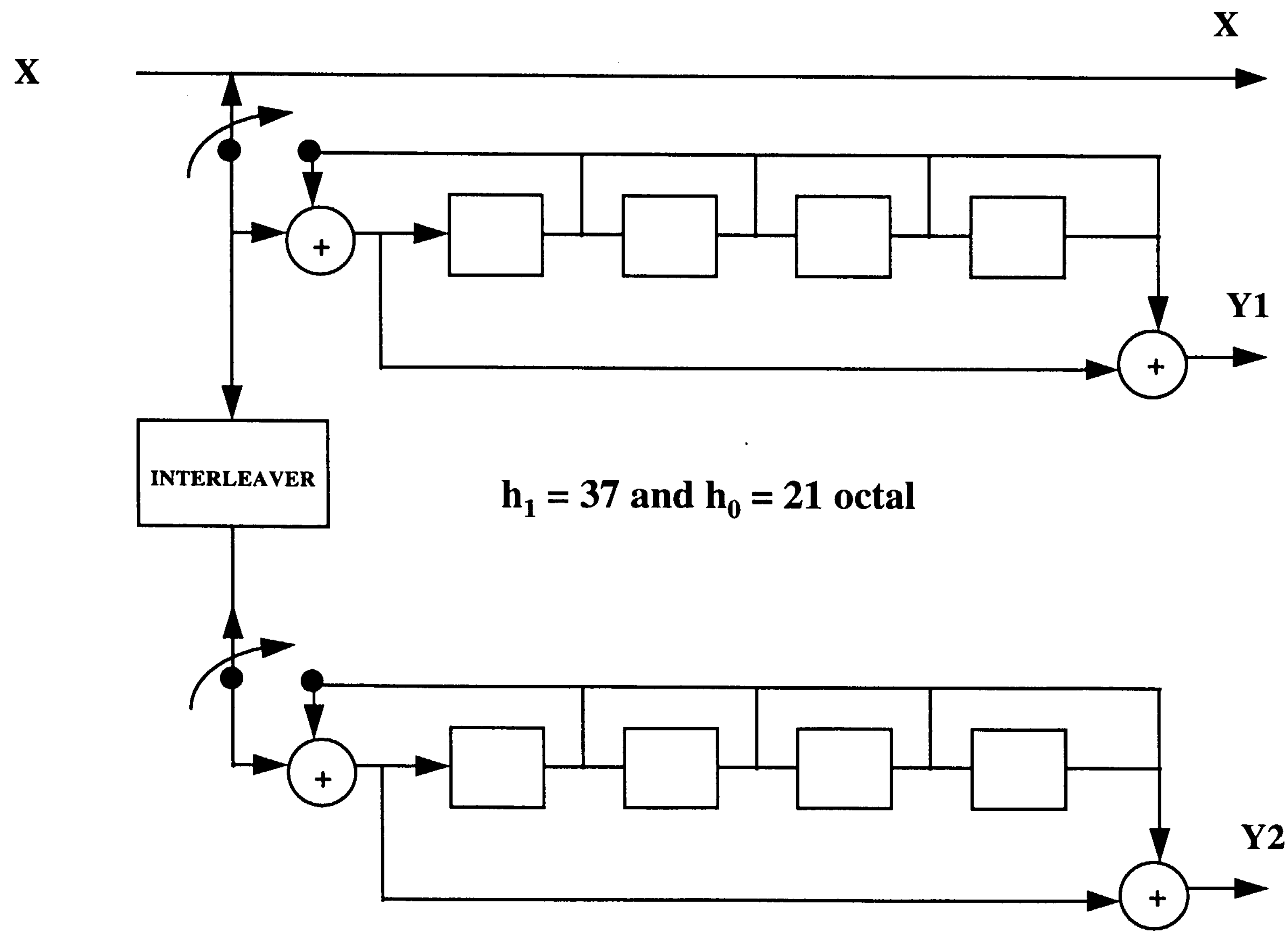


Turbo Decoder Structure

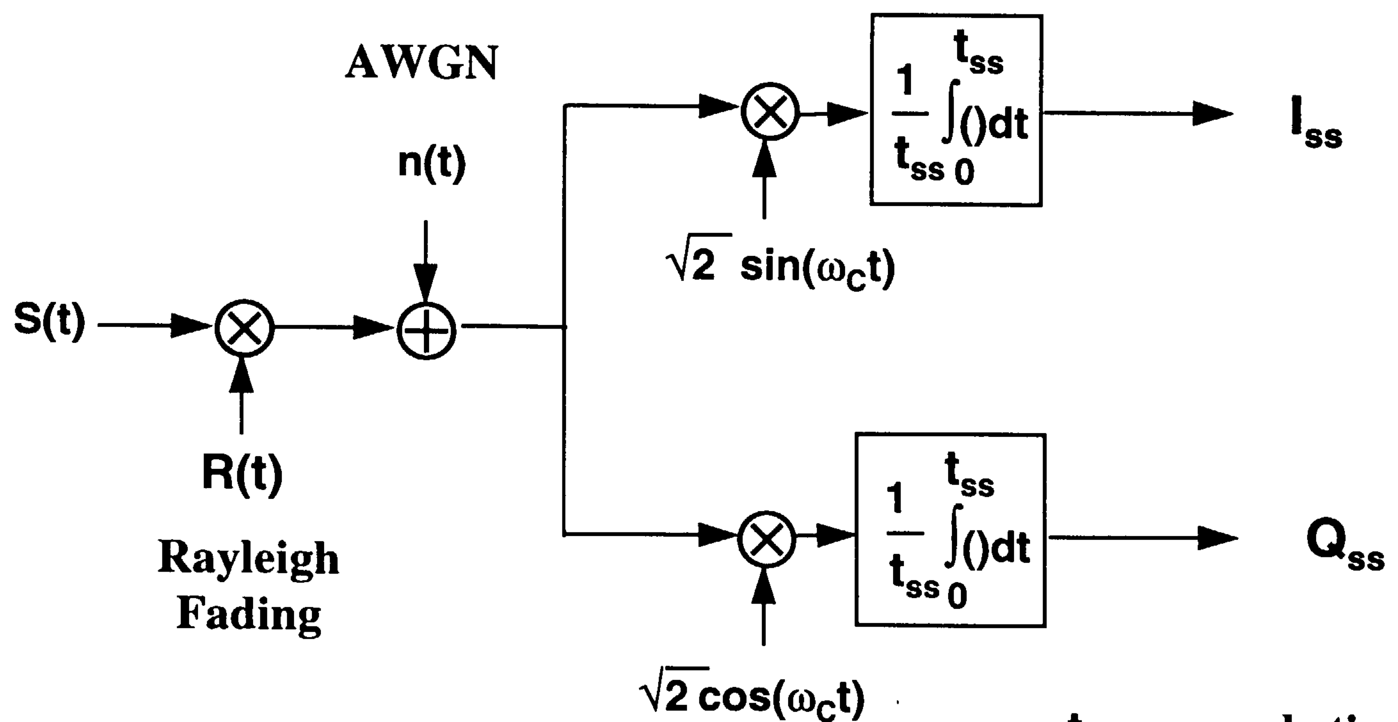


SOVA = Soft Output Viterbi Algorithm
MLSG = Maximal Length Sequence Generator

R=1/3 PCCC Showing Code Termination



Receiver Structure and Channel Model



t_{ss} = sample time = $T_s/8$
T_s = symbol time
E_s = energy / symbol
ω_C = radian carrier frequency
f_k = M-ary FSK symbol tone
Φ_k = M-ary DPSK symbol phase
Θ = initial phase offset

$$s(t) = \sqrt{2E_s/T_s}\,\sin(\omega_C t + 2\pi f_k t + \Phi_k + \Theta)$$

and $n(t) = \sqrt{2}\,(\,n_I(t)\sin(\omega_C t) + n_Q(t)\cos(\omega_C t)\,)$

M-ary DPSK Modulation / Demodulation

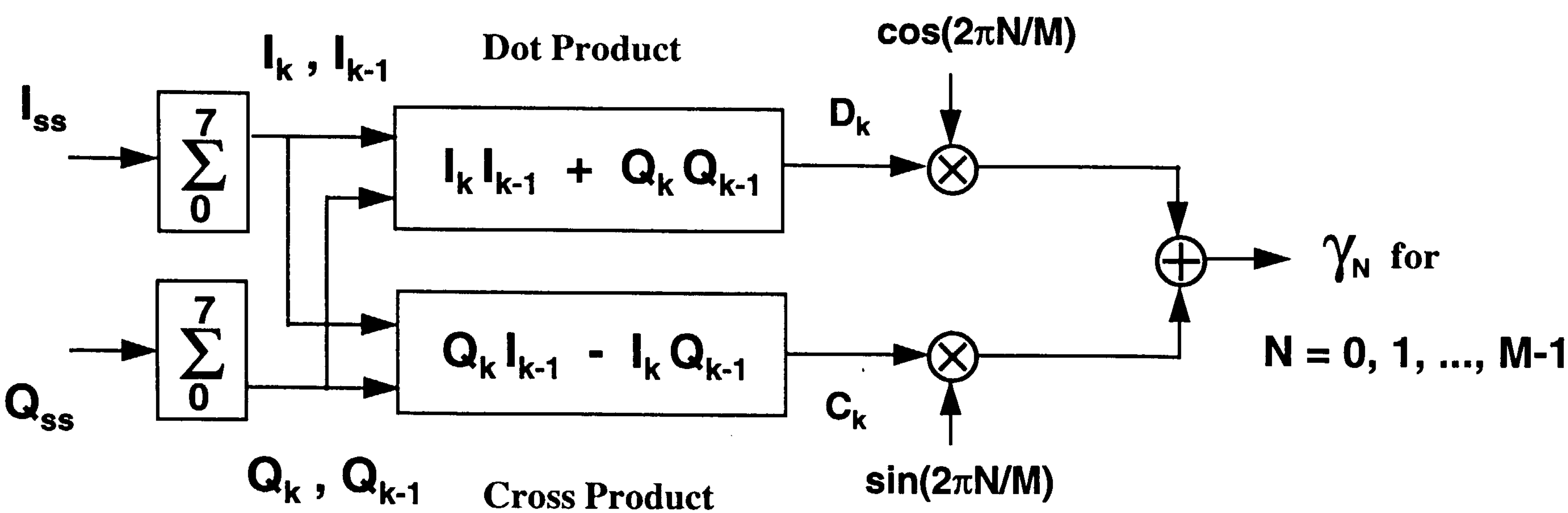
M-ary DPSK Modulation ($f_k = 0$)

$s(t) = \sqrt{2E_s/T_s}\,\sin(\omega_C t + \Phi_k + \Theta)$, $\Phi_k = \Phi_{k-1} + N\,2\pi / M$ where

k = symbol time, $k = 0, 1, 2, \ldots$

N = transmitted symbol, $N = 0, 1, \ldots, M-1$

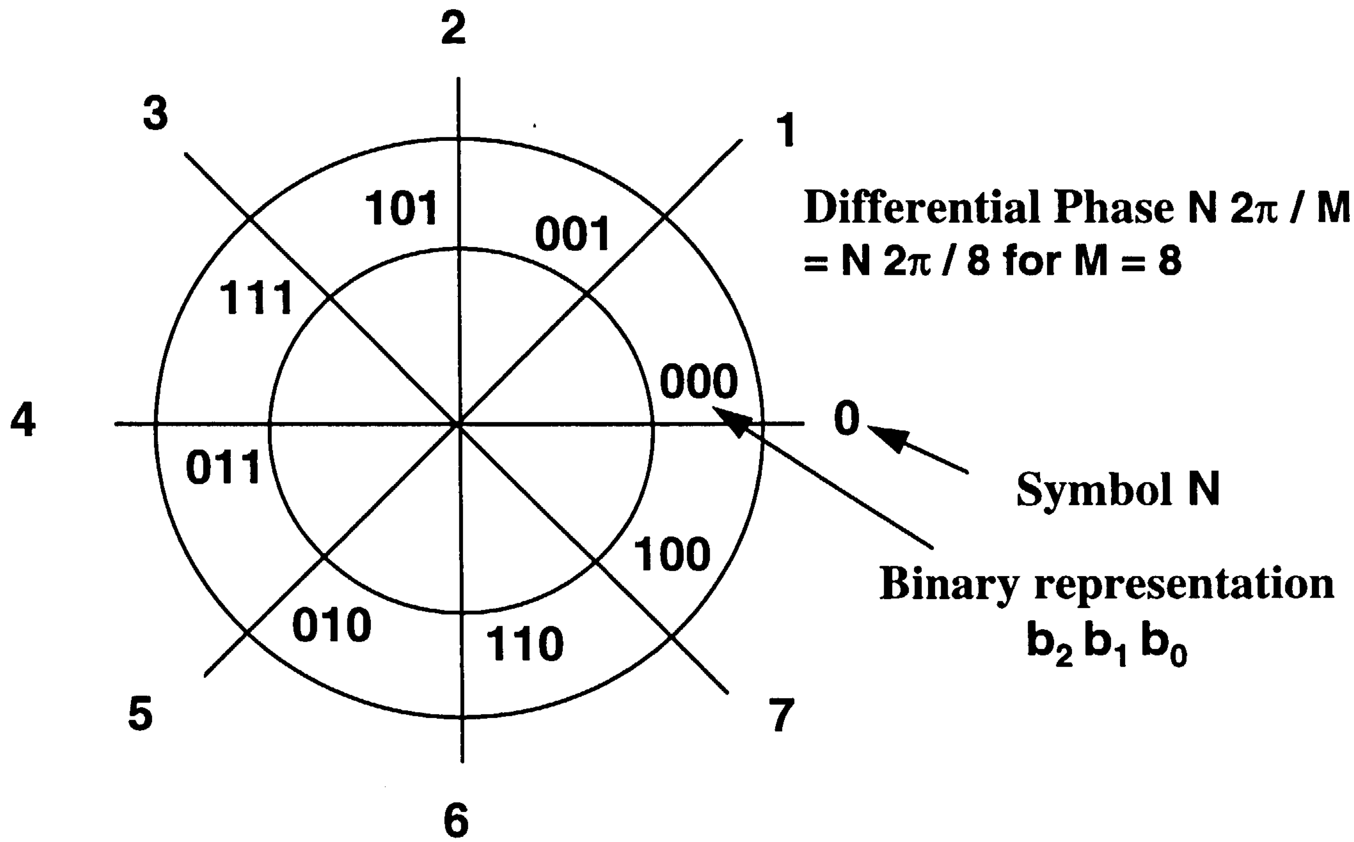
M-ary DPSK Demodulation



M-ary DPSK Demodulated
Symbol to Binary Soft Data Mapping

Coded bits $b_{p-1}..b_0$ where $p = \log_2(M)$ are mapped to symbol N using Gray encoding

M = 8 Example



Soft Binary Data

$b_2 = \mathrm{Max}(\gamma_0, \gamma_1, \gamma_4, \gamma_5) - \mathrm{Max}(\gamma_2, \gamma_3, \gamma_6, \gamma_7)$

$b_1 = \mathrm{Max}(\gamma_0, \gamma_1, \gamma_2, \gamma_7) - \mathrm{Max}(\gamma_3, \gamma_4, \gamma_5, \gamma_6)$

$b_0 = \mathrm{Max}(\gamma_0, \gamma_5, \gamma_6, \gamma_7) - \mathrm{Max}(\gamma_1, \gamma_2, \gamma_3, \gamma_4)$

M-ary FSK Modulation / Demodulation

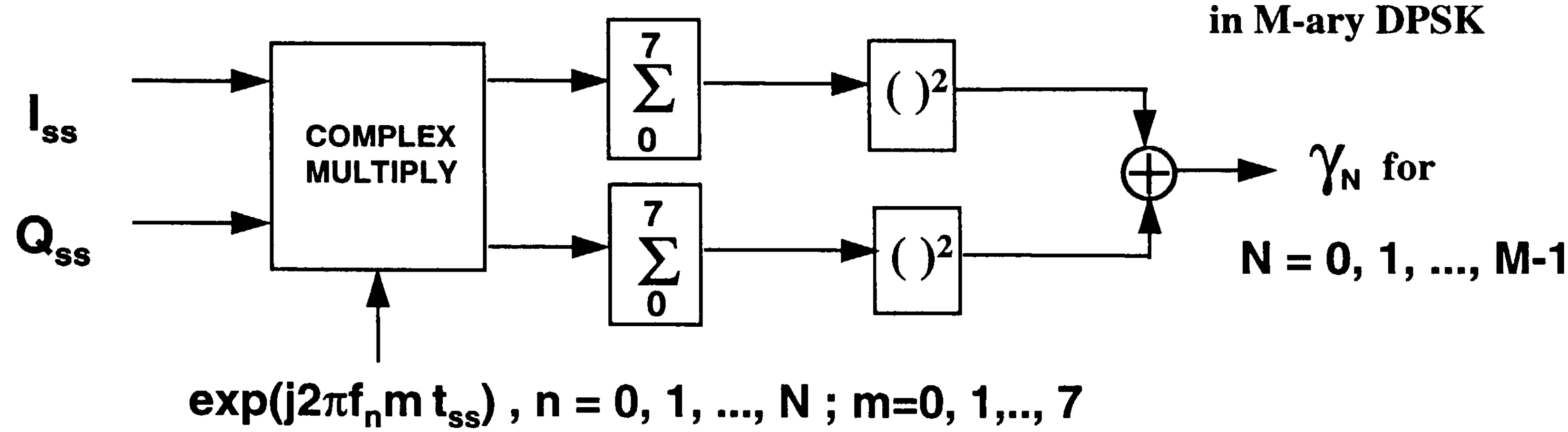
M-ary FSK Modulation ($\Phi_k = 0$)

$$s(t) = \sqrt{2E_s/T_s}\,\sin(\omega_C t + 2\pi f_k t + \Theta)\ ,\quad f_k = -(M-1)(1/(2T_s)) + N(1/T_s)$$

k = symbol time, $k = 0, 1, 2, \ldots.$
N = transmitted symbol, $N = 0, 1, \ldots, M-1$

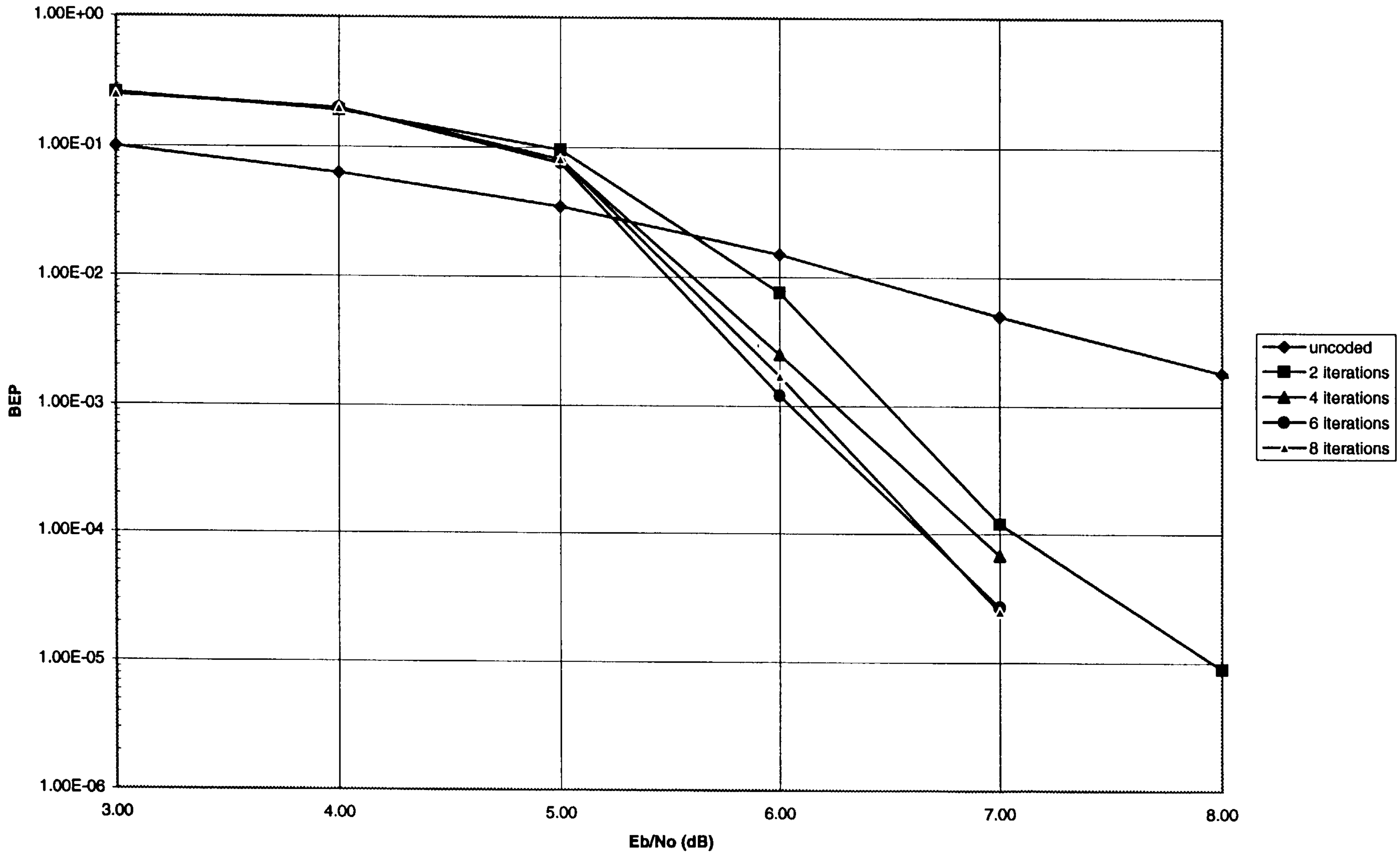
M-ary FSK Demodulation

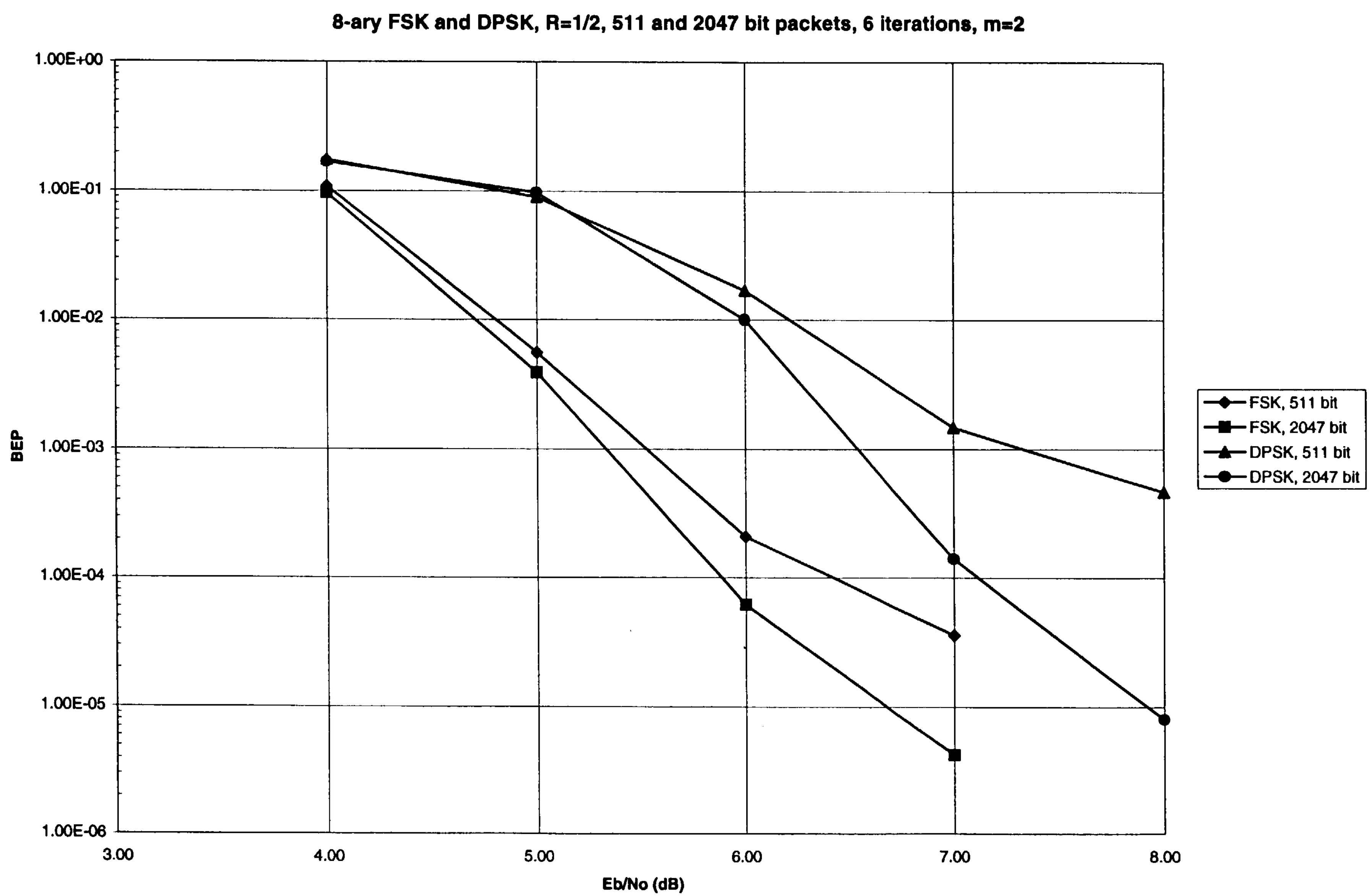
The γ_N are mapped to soft binary data as in M-ary DPSK

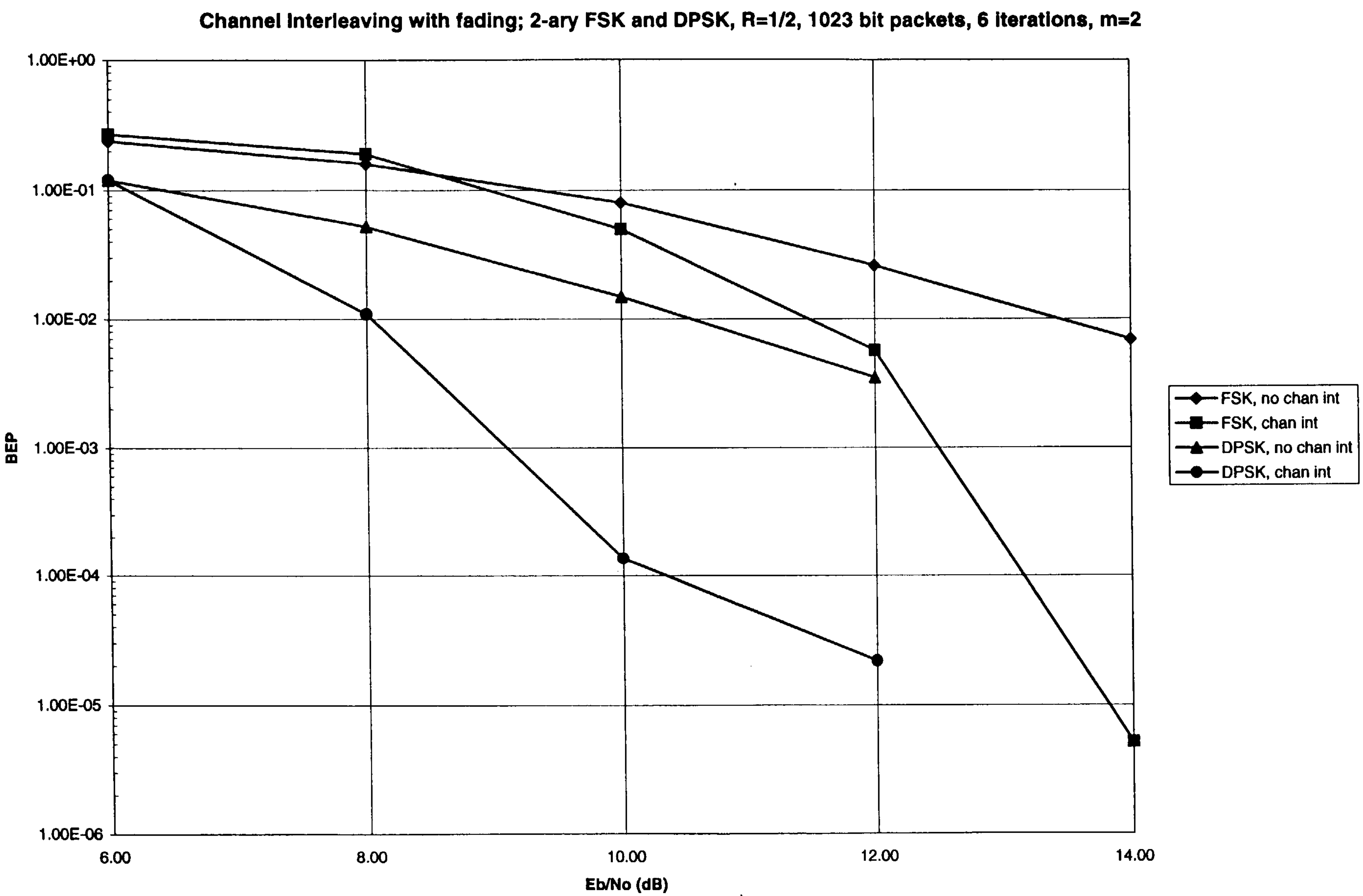


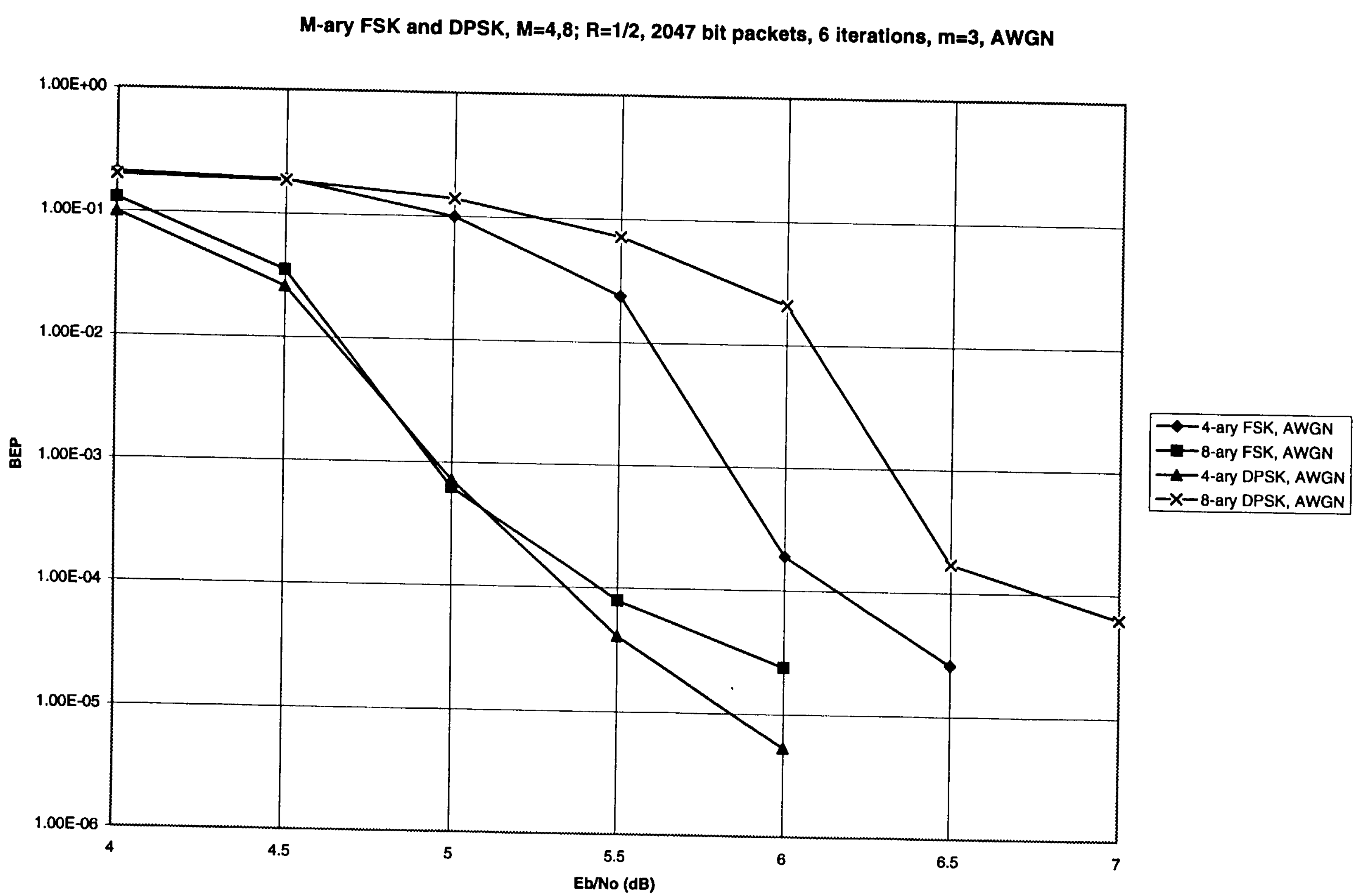
f_n = Orthogonal frequencies for $n = 0,1,\ldots,M-1$
t_{ss} = sample time = $T_s/8$

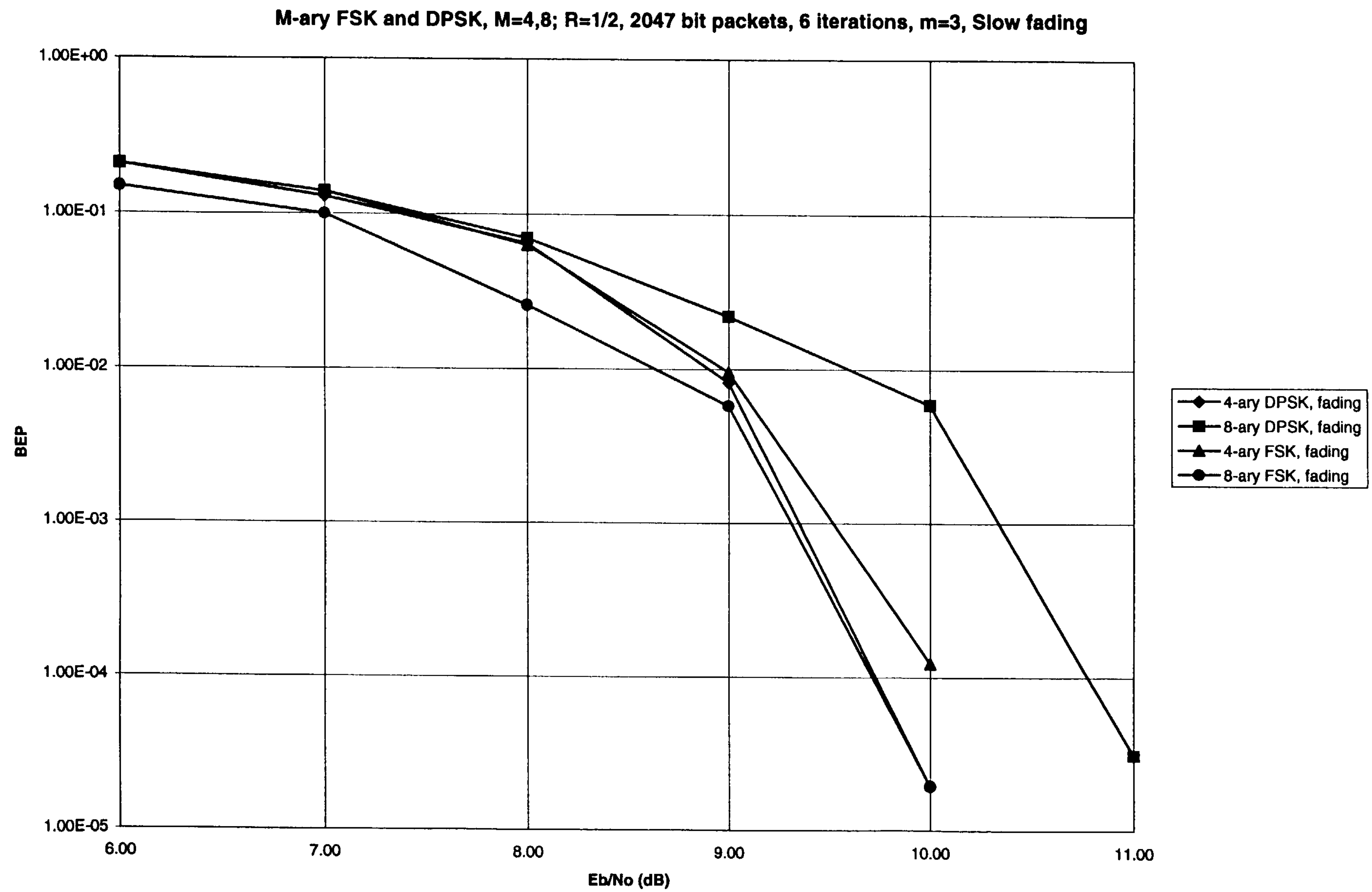
4-ary FSK, R=1/2, 1023 bit packet, m=2; 2,4,6,8 iterations




8-ary FSK and DPSK, R=1/2, 511 and 2047 bit packets, 6 iterations, m=2
BEP
1.00E+00
1.00E-01
1.00E-02
1.00E-03
1.00E-04
1.00E-05
1.00E-06
3.00
4.00
5.00
6.00
7.00
8.00
Eb/No (dB)
FSK, 511 bit
FSK, 2047 bit
DPSK, 511 bit
DPSK, 2047 bit


Channel Interleaving with fading; 2-ary FSK and DPSK, R=1/2, 1023 bit packets, 6 iterations, m=2
BEP
1.00E+00
1.00E-01
1.00E-02
1.00E-03
1.00E-04
1.00E-05
1.00E-06
6.00
8.00
10.00
12.00
14.00
Eb/No (dB)
FSK, no chan int
FSK, chan int
DPSK, no chan int
DPSK, chan int


M-ary FSK and DPSK, M=4,8; R=1/2, 2047 bit packets, 6 iterations, m=3, AWGN
BEP
1.00E+00
1.00E-01
1.00E-02
1.00E-03
1.00E-04
1.00E-05
1.00E-06
4
4.5
5
5.5
6
6.5
7
Eb/No (dB)
4-ary FSK, AWGN
8-ary FSK, AWGN
4-ary DPSK, AWGN
8-ary DPSK, AWGN


M-ary FSK and DPSK, M=4,8; R=1/2, 2047 bit packets, 6 iterations, m=3, Slow fading
BEP
1.00E+00
1.00E-01
1.00E-02
1.00E-03
1.00E-04
1.00E-05
6.00
7.00
8.00
9.00
10.00
11.00
Eb/No (dB)
4-ary DPSK, fading
8-ary DPSK, fading
4-ary FSK, fading
8-ary FSK, fading

Conclusions

- Turbo coding using iterative SOVA decoding and M-ary differentially coherent or non-coherent modulation can provide an effective coding / modulation solution
    - Energy efficient with relatively simple SOVA decoding and small packet lengths, depending on BEP required
    - Low number of decoding iterations required
    - Robustness in fading with channel interleaving

Future Investigations

- Apply Trellis Coding in addition to Turbo Coding
    - Decoding operates on soft symbols versus soft binary data
- Use $R = 2/3$ constituent codes versus punctured $R = 1/2$ codes
    - Can provide larger d_2 and d_{free}
- Formulate error probability bounds for AWGN and fading
- SOVA performance at low signal / noise ratios
- Use of simplified MAP decoding versus SOVA